Fast Multi-Level Co-Clustering

by

Haifeng Xu

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Hans De Sterck

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

We present a new multilevel method for hierarchical co-clustering. The fast multilevel co-clustering method (FMCC) implements a bi-coarsening process on the bipartite graph induced by the feature matrix. It does so in a recursive manner, producing a hierarchy of overlapping co-clusters and their connections to each other, as encoded in the co-cluster membership matrices and the coarse feature matrices that are obtained in the graph coarsening procedure. FMCC is inspired by principles of the algebraic multigrid (AMG) method for solving linear equation systems, which uses heuristic grouping criteria that are based on strength of connection in the operator matrix and are fast and scalable. Compared with other co-clustering algorithms, FMCC has the following advantages: it is computationally efficient (almost *linear* in the data size); there is no need to specify the number of clusters since FMCC finds it automatically; and the clustering gives hierarchical structure for both the row and the column variables. FMCC produces interpretable co-clusters on several recursive levels along with information on how they are connected, and thus allows to investigate the potential multilevel co-cluster structure of complex real data. The method is accurate, fast and scalable, as demonstrated by numerical tests on co-clustering problems with synthetic and real data from the fields of gene expression data and online social networks.

Acknowledgements

I would like to thank all the little people who made this possible.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Clustering is a classification problem which seeks to group data points according to the proximities between point pairs. These problems are prevalent from finding groups of friends in daily life to classifying the categories of massive online information. Though it is not hard for humans to detect the communities of small data sets, clustering is far from trivial for computers and for large-scale data sets. This research paper is on the algorithms for clustering.

Generally, there are two kinds of clustering problems according to the difference of data types. One kind can be described by a graph in which each node corresponds to a data point and the adjacency matrix describes the proximities between node pairs. Then a clustering seeks to find the node groups that have strong internal connections but sparse external ones. Researchers have proposed different measures, such as modularity [25], conductance[15] and saliency[18], based on the original graph to judge how community-like these groups are. As a result, they can develop algorithms to find proper clustering simply by optimizing those measures.

Another line of clustering problems is given in the form of data points with corresponding feature vectors , which is also called point-feature matrix. For example, the document-word data use each row to describe the frequency of each word, and clustering seeks to group the documents using their feature vectors. This kind of data can be reduced to the above one by computing point pair proximities from the feature matrix. However, the point-feature data can further have their own clustering algorithms. For example, the well-known K-means algorithm [11] seeks K centers among those data points that minimize the square sum of the Euclidean distances from the points' feature vectors to their nearest centers.

Modern development has imposed more difficulties on clustering. Besides accuracy, many

applications are seeking richer clustering structures; for example, some data can be described by the hierarchical clustering structures, but gene expression data prefer bi-clustering (or co-clustering). In addition, given that the K-means is NP-hard [1] and those optimization-based algorithms could also have high time cost, researchers are developing highly efficient algorithms to deal with massive Internet information. As a result of the above facts, researchers have shown an explosion of interest in clustering and have developed many different kinds of clustering algorithms.

1.1 Clustering Methods

In this chapter, we introduce and briefly compare different kinds of clustering methods.

1.1.1 Hard versus Soft

A hard clustering is a partition of data points, that is, every data point entirely belongs to exactly one cluster. Soft, or overlapping, clustering can assign one point to different clusters. Fuzzy clustering is a special kind of soft clustering that assigns a point to different clusters with different membership intensities (usually sum up to 1). Fuzzy clustering can be converted to a hard one by simply assigning the points to the groups in which they have maximum memberships.

1.1.2 Partitional versus Hierarchical

A partitional clustering method finds clusters at a single level. For example, it groups the faculty at a university into two clusters, i.e., the Art faculty and Science faculty. A hierarchical clustering is a nested sequence of clustering which finds the multi-level memberships between clusters at different levels. For example, a hierarchical clustering would further find the Mathematics, Physics and Computer Science faculties within the above Science faculty, and find the Applied Math, Pure Math faculties within the above Mathematics faculty, and so on. Finally, a clustering tree is produced to describe the relations between clusters at neighboring levels.

Generally, there are two kinds of hierarchical clustering, namely divisive and agglomerative hierarchical clustering. A divisive hierarchical clustering method starts from the whole data set and classifies it into big groups, then divides the big groups into smaller ones, and so

on. Conversely, an agglomerative hierarchical clustering starts from individual data points and merges the closest points iteratively into a group, which will be regarded as a new individual point at higher level. During this process, the algorithm needs a proper way to define the proximity between the original points and the new-generated points, i.e., the groups of points.

1.1.3 One-side clustering versus Co-clustering

Assume we are given a data set with points and their feature intensities (usually in the form of a matrix F). One-side clustering finds the clusters only for the points, while co-clustering (or bi-clustering) finds the clusters for both the points and features simultaneously. Note that if the data only has points and their proximities, it can only use one-side clustering.

Bioinformatics and text data mining are two popular applications for co-clustering. Many co-clustering algorithms have been developed to classify the gene expression data in bioinformatics. Prelić *et al* [28] gives a systematic comparison of some well-known co-clustering algorithms, such as Cheng and Church's block clustering algorithm [6] and the order preserving submatrix algorithm, OPSM [2]. In text data mining, researchers are interested in finding the clusters for both documents and words by co-clustering algorithms, for example, the bipartite graph partition algorithm in [8] and the information-theoretic algorithm in [9]. The non-negative matrix factorization (NMF), a recently proposed method, also shows its competitive ability in co-clustering for both gene expression data [5, 16] and text data [38].

1.2 Challenges in Clustering

Although well developed, clustering is still an explosive research field with many challenging problems.

One of the challenges in clustering problems is to compare and measure clustering results. Since the actual classes of data points (usually in very large size) are not known, there is not a way to compare the results with ground truth or that of the other algorithms using real data sets. To show the algorithm performances most papers have to test their algorithms on artificial data or some standard data sets with human-generated "ground truth".

Another challenge is to find the number of clusters. Most of the powerful clustering algorithms need to specify a proper number of clusters K beforehand, e.g., the K-means, NMF

and information-theoretic algorithm in [9]. However, given a initial data set, it is far from trivial to decide the proper K due to a lack of prior knowledge of the data. Actually, there is a whole field of research focusing on finding a proper K. One may refer to [26] for an overview on different methods.

Finally, some difficulties are posed by the wide spread applications and large online data sets. The main problem of clustering algorithms is that the cannot be standardized. Algorithm developed may give best result with one type of data set but may fail or give poor results with data set of other types. This limits the spread of clustering algorithms in different applications. In addition, some sophisticated clustering algorithms have high computational complexity, e.g., K-means and NMF. This limits their applications in the large scale online data which unfortunately happens to be the main application area of clustering. So efficient and scalable clustering algorithms, even with a little loss in accuracy, are becoming more attractive in modern applications.

1.3 Overview

This research paper considers the following problem:

Problem. Let $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_n\}$ be two different element sets. $F \in R^{m \times n}$ is the feature intensity matrix, which captures the relations between X and Y. For simplicity, we call F the $x - y$ feature matrix. For example, X, Y could be the document and word set, respectively, and f_{ij}, an entry of F, describes the frequency of the word y_j in the document x_i; or X, Y could be the LinkedIn user and the professional skill set, respectively, and f_{ij} is the intensity at which user x_i masters skill y_j. We are interested in simultaneously find the hierarchical clustering of X and Y based on F.

We propose a hierarchical co-clustering framework that is inspired by methods of Algebraic Multigrid (AMG). AMG was originally developed as an iterative solution method for linear systems that arise from discretizing elliptic Partial Differential Equation (PDE) problems on unstructured grids [3, 4]. Discretized PDE matrices on unstructured grids can be interpreted as weighted graph matrices (with the graph edges corresponding to the grid edges), and this analogy opens the door to applying ideas from multilevel numerical methods for PDEs on unstructured grids to graph problems. In the context of AMG, this was first done by Brandt and co-workers for problems in image segmentation [30, 31] and in clustering and manifold detection [19]. In this approach, components from the standard AMG algorithm that include coarsening and interpolation based on strength of connection

in the operator matrix, and so-called variational coarse operator definition [3, 4], can be applied directly to graph problems to create hierarchical overlapping groupings of graph nodes (nested image segments or graph clusters) in a graph coarsening process, and to detect salient groupings. These clustering methods, which are based on heuristic strength-based grouping criteria that are fast and scalable, have proven highly attractive due to their speed and inherent scalability, and ability to reveal multilevel structure, which is used to improve accuracy for difficult segmentation and clustering problems [19, 30, 31]. It would be desirable to develop AMG-inspired algorithms for co-clustering that enjoy these beneficial properties, but applying AMG principles to co-clustering incurs new challenges and has not been attempted yet.

This research paper formulates new AMG-inspired multilevel methods for hierarchical co-clustering based on heuristic strength-based grouping criteria that are fast and scalable. When applying AMG principles to the co-clustering problem, a first important challenge that needs to be addressed is to find proper ways to define coarse/fine (C/F) splittings for the rectangular feature matrices F, in which both the columns and rows of F are coarsened. Here, the C-points are seed points for clusters on the next coarser level, and the F-points belong to C-clusters in an overlapping fashion based on their connection strength to the C-points in the graph. We propose two approaches. The first approach effectively *approximates* the products FF^T and compute correlations (or distances) between the row variables, and F^TF to compute correlations between the column variables, and then coarsens the row and column variables separately based on these correlation strengths using standard AMG C/F-splitting techniques [3, 4]. A similar approach was used in [35] to coarsen rectangular matrices for computing the singular values and singular vectors of rectangular matrices using AMG techniques. The second approach for C/F-splitting we propose is novel: C/F-point selection alternates between row and column variables, based on the connection strength between row and column variables as encoded directly in the feature matrix F. It can be seen as a coarsening process on the bipartite graph induced by the feature matrix F. This approach avoids the (costly) computation of FF^T and F^TF, and instead groups row variables indirectly based on their connections to column variables (and vice versa). This novel method has great potential significance: it directly uses the correlation information encoded in the feature matrix F, and is thus faster and more scalable than basing cluster grouping on computation of FF^T and F^TF, which makes it an attractive candidate coarsening algorithm for large co-clustering problems. A second challenge in applying AMG principles to co-clustering lies in properly defining interpolation matrices and coarse feature matrices. Here also we consider two alternatives: the first is based on row-column and column-row interpolation, and the second is based on row-row and column-column interpolation. Further challenges we address include issues of scaling

and normalization. The resulting novel fast multilevel co-clustering (FMCC) algorithm produces a multilevel hierarchy of overlapping co-clusters and their connections to each other as encoded in coarse feature matrices. It is important to note that the method produces interpretable co-clusters on many recursive levels along with information on how they are connected, and thus allows to investigate the potential multilevel co-cluster structure of complex real data. The method is accurate, fast and scalable, as demonstrated by numerical tests on co-clustering problems with synthetic and real data from the fields of gene expression data and online social networks.

The rest of the paper is organized as follows. Chapter 2 gives the description of the novel framework of hierarchical co-clustering. Chapter 3 lists some details in the algorithms of FMCC. We analyze the computational complexity of FMCC in Chapter 4. The systematical experimental tests are reported in Chapter 5. We conclude the paper in Chapter 6, while leave a brief review of clustering measures to the Appendix.

Chapter 2

The Fast Multi-Level Co-Clustering Framework

In this chapter, we formulate a novel framework called Fast Multi-Level Co-Clustering (FMCC) that is inspired by Algebraic Multi-Grid (AMG). We assume that F is the only given information, and all the rows and columns of F have at least one non-zero entry. For discrimination, we call the points in X x-points, and y-points in Y. We are interested in the hierarchical clustering for both X and Y.

2.1 Basic Ideas

One may imagine the x-points and y-points as nodes in a graph. A key observation here is that any entry $f_{i,j}$ of F can be viewed as the connection strength between x_i and y_j. So the following $(m+n) \times (m+n)$ symmetric matrix captures the connection strength between all the points in X and Y:

$$A = \begin{bmatrix} 0_{m\times m} & F_{m\times n} \\ (F^T)_{n\times m} & 0_{n\times n} \end{bmatrix} \tag{2.1}$$

in which the first m rows (or columns) correspond to x-points while the other n rows correspond to y-points.

The matrix in 2.1 shows that the x-points do not have explicit connections to other x-points, neither do y-points. So A can be viewed as a weighted adjacency matrix of a bipartite

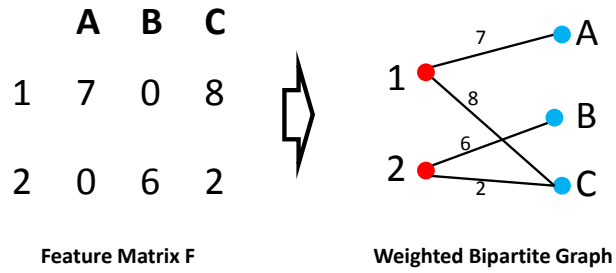

Figure 2.1.1: From the feature matrix F to the weighted bipartite graph

graph with the set X and Y as two parts. Actually, any feature matrix F describes a weighted bipartite graph; an illustrative example is given in Figure 2.1.1.

From this perspective, a simultaneous clustering of X and Y can be viewed as a clustering of the weighted bipartite graph induced by F. The conflict here is that we need the connection relations within x-points or y-points, but only the connection strengths between x-points and y-points are known. So we have to uncover the x-point relations according to their interactions with y-points and vice versa. In addition, to find the multi-level clustering structure, we need to construct properly the feature matrices on the coarse levels and do clustering iteratively.

As a result, the hierarchical co-clustering framework will execute the following two phases iteratively. Given the level r adjacency matrix $A^{[r]}$,

- C,F-Coarsening Phase: find the "representative" x-points and y-points at level r, which we call C-points (i.e., coarse points);
- Interpolation Phase: construct the interpolation matrices, coarse adjacency matrix $A^{[r+1]}$ as well as the membership matrices between fine and coarse points.

Each C-point could be viewed as a representative of (but not equivalent to) a clustering group that becomes a seed point in the coarse level. All the points left become F-points (i.e., fine points) whose connections are merged into the coarse-level seed points by weighted aggregation. The seed points at each level, together with the membership matrices between them, give the hierarchical clustering.

Before going to the detailed descriptions of these two phases, we first describe some notations. Without loss of generality, we take a virtual angle of the current level and aim at the construction of the coarse (next) level.

Notations: C_x and F_x are the sets of x-points that are C-points and F-points, respectively. While x_c/x_f denotes an individual x-point that is C/F-point. Similarly, C_y/F_y is the C/F-coarsening for the y-points, while y_c/y_f denotes an individual y-point. F is the $x-y$ feature matrix and G_F is the bipartite graph induced by matrix F. A is the adjacency matrix of G_F. M_x and M_y are the membership matrices for x-points and y-points, respectively. F^c and A^c are the $x-y$ feature matrix and adjacency matrix at coarse level, respectively. f_{ij}, F_i, F^j denote the (i,j) entry, i'th row, j'th column of F, respectively.

2.2 C,F-Coarsening Phase

The C,F-coarsening phase aims to find the representatives of the clusters that become seed points at coarse level. Our construction starts from the concept of "strong connection".

Definition 1. Given a threshold θ, for any $x_i \in X$ and $y_j \in Y$, x_i is strongly connected to y_j if either $f_{ij} \geq \theta(\max_k f_{kj})$ *or* $f_{ij} \geq \theta(\max_k f_{ik})$.

We use a binary strength matrix $S \in \{0,1\}^{m\times n}$ to record the strong connections, in which $S_{ij} = 1$ means x_i is strongly connected to y_j, and zero otherwise. Viewed from the perspective of graph, S defines an unweighted bipartite graph G_S by deleting the "weak" connections in G_F. We have the following proposition from definition 1.

Proposition 2. *If all the rows and columns of F have at least one non-zero entry, then Definition 1 guarantees that each x-point (or y-point) is strongly connected to at least one y-point (or x-point), i.e., G_S is connected.*

From now on, let's think about the set X and Y in a bipartite graph, in which any edge indicates a *strong* connection. Based on this view, we propose two different algorithms to uncover the C-points.

Perhaps, the most natural way for coarsening is to compute the products FF^T and F^TF as the distances between x-points and y-points, respectively. Then one can use the standard AMG C,F-splitting techniques like [30, 31]. However we note that the directly computed inner product distances have several drawbacks. Let's assume for illustration a special case of three x-points with noisy features $F_1 = (10,10,10,1,1,1,...,1) \in R^{1000}$, $F_2 = (10,10,10,0,0,0,...,0) \in R^{1000}$ and $F_3 = (1,1,1,10,10,10,1,...,1) \in R^{1000}$ (those 1's are noises). Then the inner product gives $(F_1,F_2) = 300$ and $(F_1,F_3) = 1030$. However, as distances, these computed inner products are not proper because F_1 shares all its strongly connected features with, therefore should be closer to, F_2, but the inner products are

Algorithm 2.1 Separate C,F-splitting

Input: S, position parameter $\rho \in [0,1]$, overlap ratio $\alpha_x, \alpha_y \in (0,1]$
Output: C_x, F_x, C_y, F_y

1. Create the increasing ordered lists for both x, y-points according to how many strong connections they have.
2. Initialize all x-points to be unassigned.
3. Choose an x-point at position ρ in the unassigned and ordered list, say x_c, and assign x_c to C_x; for any unassigned point x_i, if $\frac{(S_i, S_{x_c})}{\sum_j S_{ij}} \geq \alpha_x$, assign x_i to F_x.

   // "at position ρ" here means at position $\lceil \rho \times$ length of the list$\rceil$ of the list.
4. Repeat step 3 until all the x-points are assigned.
5. Repeat steps 2 to 4 for y-points.
6. Output C_x, F_x, C_y and F_y.

dominated by the noise unfortunately. What's more, this computation is expensive because (F_1, F_3) needs 1000 multiplications, most of which are brought by noises.

As a result, instead of looking at $F^T F$ and $F^T F$, we turn to the products SS^T and $S^T S$. Recalling that S is binary and is sparser than F, so its multiplications are cheaper than those of F. More importantly, the inner product (S_i, S_j) has a proper interpretation, which is the number of shared strongly connected features between x_i and x_j, and can better describe the proximity. These inspire our design of Algorithm 2.1, the Separate C,F-Splitting algorithm.

Further explanations of Algorithm 2.1:

1. $\frac{(S_i, S_{x_c})}{\sum_j S_{ij}} \geq \alpha$ in step 3 means that x_i shares more than α of its strongly connected features with x_c. Note that $\sum_j S_{ij}$ has already been computed in step 1, and (S_i, S_{x_c}) can be computed in advance by the multiplication SS^T.
2. The parameter ρ properly decides which C-point should be chosen as the representative at each step. When $\rho = 1$, the point with most connections is picked as C-point. But we should notice that the point with most connections is not necessarily representative in practice, for example, some popular words (e.g., "the", "is") in document

clustering, or some popular skills (e.g., "C", "Matlab") in the clustering of people's professions.

Algorithm 2.1 shares similar intuition from the standard AMG C,F-splitting techniques except that it has a more effective construction of the distances of point pairs. We now describe a novel C,F-splitting algorithm, namely Alternating C,F-Splitting (Algorithm 2.2), which is new even to the literature of the AMG-based algorithms. This algorithm avoids the costly computation of distances (i.e., the inner products) within X or Y, instead we directly use the $x-y$ point connections encoded in the feature matrix F and decide similar x-points according to their interactions with y-points and vice versa.

The Alternating C,F-Splitting algorithm is inspired by the following intuition: if x_i is strongly connected to y_j, and y_j strongly connects several other x-points $\{x_{i_1}, ..., x_{i_k}\}$, then $\{x_i, x_{i_1}, ..., x_{i_k}\}$ might be in the same group (because they share a strongly connected feature y_j). Similarly, all the other y-points that strongly connects x_i, say $\{y_{j_1}, y_{j_2}, ..., y_{j_s}\}$, might be in the same cluster as y_j. As a result, only one of those x-points (or y-point) should be a C-point.

Further explanations of Algorithm 2.2:

1. The algorithm executes steps 2,3 and 4 in an alternating fashion, in order to balance the numbers of coarse x-points and coarse y-points. This is why it is called "alternating C,F-splitting".

2. Step 3 is executed to guarantee that each coarse x-point strongly connects to at least one coarse y-point. We will show later that this is crucial to guarantee proper constructions of membership matrices and F^c.

Algorithm 2.1 construct C_x and C_y separately, while Algorithm 2.2 has an interactive construction. The key difference between these two algorithms lies in the usage of information. In particular, algorithm 2.2 decides similarity based on only one shared feature, while algorithm 2.1 considers all the shared features between each point pair. As a result, Algorithm 2.2 is expected to have lower complexity but induce less accurate matches, which is also proved by our later complexity analysis and experiments.

2.3 Interpolation Phase

In this section, we describe how to build the coarse adjacency matrix A^c by the AMG-like interpolations. The constructions of the feature matrix F^c as well as the membership matrices M_x and M_y will also be discussed.

Algorithm 2.2 Alternating C,F-splitting

Input: S, position parameter $\rho \in [0, 1]$
Output: C_x, F_x, C_y, F_y

1. Create the increasing ordered lists for both x-points and y-points according to how many strong connections they have. Initialize all x, y-points to be unassigned.

2. If all the x-points are assigned, go to step 4;

   Otherwise: choose the x-point at position ρ in the unassigned and ordered list, say x_c, and assign it to the set C_x; Assign all the unassigned y-points that are strongly connected to x_c to the set F_y.

   // "at position ρ" here means at position $\lceil \rho \times$ length of the list$\rceil$ of the list.

3. If x_c is not strongly connected to any $y_j \in C_y$, then pick the y-point at position ρ in the ordered list of $\{y_j | S_{x_c j} = 1\}$, say y_c. Then, re-assign y_c to C_y; Assign all the unassigned x-points that are strongly connected to y_c to the set F_x.

4. If all the y-points are assigned, go to step 5; Otherwise, repeat step 2 and 3, but exchange the role of x and y.

5. Repeat steps 2 to 4 until all the x, y-points are assigned. Output C_x, F_x, C_y and F_y.

2.3.1 Constructions of the Clustering Membership Matrices

Let $M_x \in R^{m \times m_c}$ and $M_y \in R^{n \times n_c}$ in which m_c and n_c are the numbers of coarse x, y-points, respectively.

Recalling that we use F_i and F^j to denote the row i and column j of F , so F can be written in the following forms:

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_m \end{bmatrix} = \left[F^1, \cdots, F^n\right]$$

Define the following two matrices:

1. Let $P_{x\to y} \in R^{m\times n_c}$ be the submatrix of F that consists of all the columns $F^j, \forall j \in C_y$; (2.2)

2. Let $P_{y\to x} \in R^{n\times m_c}$ be the *transpose* of the submatrix of F that consists of all the rows $F_i, \forall i \in C_x$; (2.3)

We note that $P_{y\to x}$ can be viewed differently from the following three aspects (so does $P_{x\to y}$): i. $P_{y\to x}$ is the connection strength matrix between all the y-points and those coarse x-points; i.i. $P_{y\to x}$ is the adjacency matrix of the bipartite graph which has C_x and Y as two parts; i.i.i. last but not least, $(P_{y\to x})^T$ is the feature matrix of coarse x-points.

Since we have the feature vectors of all the points, naturally, the basic idea to construct membership matrices is to compute the distances between the F-points and C-points. In principle, any proper metric could be a choice. We choose the inner product due to its simplicity and low-cost computation. This results in the following simple construction forms of M_x and M_y:

$$M_x:\ \text{i. } M_x = FP_{y\to x};\quad \text{ii. linearly normalize each row of } M_x \text{to sum up to 1}$$

$$M_y:\ \text{i. } M_y = F^TP_{x\to y};\quad \text{ii. linearly normalize each row of } M_y \text{to sum up to 1}$$

Here all the inner product computations are combined together as a matrix multiplication; normalization gives the probabilities (sum to 1) that a fine point belongs to each C-point.

One potential problem during the construction is that M_x or M_y might have rows in which all the elements are zero. This is an ill case which means a fine point does not belong to any cluster (i.e., C-point). Fortunately, we can show that both Separate and Alternating C,F-splitting can guarantee that M_x and M_y will not suffer from such ill case.

Theorem 3. *If all the rows and columns of F have non-zeros, then all the rows and columns of M_x or M_y have non-zeros under Separate C,F-Coarsening (Algorithm 2.1).*

Proof. We only show the conclusion for M_x. The reasoning for M_y is similar.

Without loss of generality, we consider the initial construction of $M_x = FP_{f\to x}$ without normalization. For any $x_i \in X$, if x_i is a C-point, then $(M_x)_{i,i} = F_iF_i^T \neq 0$ because the

row vector F_i has non-zeros according to the assumption, therefore the i'th row as well as its corresponding column of C-points have non-zeros; otherwise, x_i is a F-point, and Algorithm 2.1 guarantees that there exists a coarse x-point x_c such that $\frac{(S_i,S_{x_c})}{\sum_j S_{ij}} \geq \alpha_x > 0$. So $(M_x)_{i,x_c} = (F_i, F_{x_c}) > 0$ will be a nonzero term in the i'th row of M_x. □

Similarly, we have the following theorem for Algorithm 2.2.

Theorem 4. *If all the rows and columns of F have non-zeros, then all the rows and columns of M_x or M_y have non-zeros under Alternating C,F-Coarsening (Algorithm 2.2).*

Proof. We still only prove for M_x and assume M_x has not been normalized.

For any $x_i \in X$, if x_i is a C-point, then $(M_x)_{i,i} = F_i F_i^T \neq 0$ because the row vector F_i has non-zeros according to the assumption, therefore the i'th row as well as its corresponding column of C-points have non-zeros; Otherwise, x_i is a F-point, proposition 2 shows that it strongly connects to at least one coarse y-point, say y_c, and the step 3 of Algorithm 2.2 guarantees that y_c strongly connects to at least one coarse x-point, say x_c, then $(M_x)_{i,x_c} = F_i F_{x_c}^T \neq 0$ because i and x_c share a common y-point, i.e., y_c. These show that any row or column of M_x has at least one non-zero. □

Note that till now, the membership matrices we constructed are between two neighboring levels. To compute the membership matrix between any two levels l_1 and l_2 ($> l_1$), we simply aggregate the memberships of these middle levels.

Definition 5. The membership matrix between two levels l_1 and l_2 ($> l_1$) is constructed as follows:

$$M^{[l_1,l_2]} = M^{[l_1,l_1+1]} ... M^{[l_2-1,l_2]} \tag{2.4}$$

in which $M^{[r,r+1]}$ denotes the membership matrix between level r and $r+1$, and M is either M_x or M_y.

One can show that if each of $M^{[r,r+1]}$ is a membership matrix, so does $M^{[l_1,l_2]}$. Specifically,

Lemma 6. *If every $M^{[r,r+1]}$ are non-negative (entry-wise) and each row sums up to 1. $M^{[l_1,l_2]}$ constructed in 2.4 is also non-negative (entry-wise) and each row sums up to 1.*

Proof. The proof is by induction. We only need to prove $M^{[r-1,r+1]} = M^{[r-1,r]}M^{[r,r+1]}$ has the desired property.

It's easy to see that any entry of $M^{[r-1,r+1]}$ is non-negative. We prove that any row of $M^{[r-1,r+1]}$ sums up to 1. Assume $M^{[r,r+1]} \in R^{m_r \times m_{r+1}}$, so $M^{[r-1,r+1]} \in R^{m_{r-1} \times m_{r+1}}$. For any row i of $M^{[r-1,r+1]}$, we compute the sum of the i'th row of $M^{[r-1,r+1]}$ as follows:

$$\begin{aligned}
\sum_{j=1}^{m_{r+1}} M_{ij}^{[r-1,r+1]} &= \sum_{j=1}^{m_{r+1}} \sum_{k=1}^{m_r} M_{ik}^{[r-1,r]} M_{kj}^{[r,r+1]} \\
&= \sum_{k=1}^{m_r} M_{ik}^{[r-1,r]} (\sum_{j=1}^{m_{r+1}} M_{kj}^{[r,r+1]}) \\
&= \sum_{k=1}^{m_r} M_{ik}^{[r-1,r]} = 1.
\end{aligned}$$

□

2.3.2 Constructions of A^c and F^c

AMG has a standard form to construct A^c, i.e., $A^c = P^T AP$ in which P is the interpolation matrix and essentially captures each point's memberships in the C-points. The rationality under this construction is to merge all the points to those C-points according to their memberships. This intuition has been used in some applications, e.g., the multi-level image segmentation [31, 30].

Our construction also follows this formulation, but we expect A^c to have some further properties in our co-clustering case. Firstly, we hope that A^c has similar anti-diagonal structure as A; and furthermore, the right-upper side sub-matrix of A^c should be able to be interpreted as the coarse feature intensity matrix.

Based on these requirements, it turns out that there are also at least two ways to construct A^c, essentially two different ways to define P.

Case 1. Anti-Diagonal P:

$$P = \begin{bmatrix} 0 & P_1 \\ P_2 & 0 \end{bmatrix} \in R^{(m+n)\times(m_c+n_c)} \tag{2.5}$$

in which $P_1 \in R^{m\times n_c}$ indicates the connection strengths between fine x-points and coarse y-points, while $P_2 \in R^{n\times m_c}$ indicates the connections between fine y-points and coarse x-points. Therefore, a natural choice of P_1 would be the matrix $P_{x\to y}$ in 2.2. Similarly, P_2 would be the matrix $P_{y\to x}$ in 2.3. In this case A^c has the form $\begin{bmatrix} 0 & F^c \\ (F^c)^T & 0 \end{bmatrix}$ in which $F^c = (P_{y\to x})^T F^T P_{x\to y}$.

Case 2. Diagonal P:

$$P = \begin{bmatrix} P_3 & 0 \\ 0 & P_4 \end{bmatrix} \in R^{(m+n)\times(m_c+n_c)} \tag{2.6}$$

in which $P_3 \in R^{m\times m_c}$ indicates the relations between fine and coarse x-points, while $P_4 \in R^{n\times n_c}$ indicates the relations between fine and coarse y-points. Therefore, a natural choice of P_3 would be M_x, while P_4 would be M_y. In this case A^c has the form $\begin{bmatrix} 0 & F^c \\ (F^c)^T & 0 \end{bmatrix}$ in which $F^c = (M_x)^T F M_y$.

Please note that in these cases we only need to build and store F^c instead of A^c in practice, which makes the computation and storage cheaper.

One important requirement during the construction is that any row or column of F^c should have non-zeros. The following theorems guarantee the validity of the above constructions.

Theorem 7. *If all the rows and columns of F have non-zeros, then all the rows and columns of $F^c = (P_{y\to x})^T F^T P_{x\to y}$ constructed in the anti-diagonal case (Case 1) also have non-zeros, if Separate/Alternating C,F-Splitting is used.*

Proof. We have shown in Theorem 3 and 4 that each row or column of $FP_{y\to x}$ has non-zeros. Note that, for any $k \le n_c$, the k'th column of $F^c = (FP_{y\to x})^T P_{x\to y}$ is a linear combination of the columns of $(FP_{y\to x})^T$ where the weights come from the k'th column of $P_{x\to y}$. Since any row or column of F has non-zeros, so the k'th column of $P_{x\to y}$ has at least one non-zero, i.e., $\exists j$, such that $(P_{x\to y})_{j,k} > 0$. Since all the entries here are non-negative, we have:

$$k'\text{th column of } F^c \ge j\text{'th column of } (FP_{y\to x})^T \times (P_{x\to y})_{j,k}$$

where "$\ge$" means entry-wise bigger or equal.

Since j'th column of $(FP_{y\to x})^T$ has non-zeros, so does k'th column of F^c. This shows that any column of F^c has non-zeros.

Similarly, one can show each row of F^c has non-zeros by viewing any row of F^c as a linear combination of the rows of $F^T P_{x\to y}$. □

Theorem 8. *If all the rows and columns of F have non-zeros, then all the rows and columns of $F^c = (M_x)^T F M_y$ constructed in the diagonal case (Case 2) also have non-zeros, if Separate/Alternating C,F-Splitting is used.*

Proof. We have shown in theorem 3 and 4 that any row or column of M_x or M_y has non-zeros. Similar to the proof of theorem 7, one can easily show any row or column of $(M_x)^T F$ has non-zeros, and so does F^c. □

Till now, we have given a detailed description about the whole picture of the novel FMCC framework. The C,F-Coarsening phase and Interpolation phase will be executed iteratively, until some stop rules are satisfied, e.g., only one point left or the number of coarse points does not decrease anymore.

Since the C,F-Coarsening phase is completely independent of the coarse level construction phase, so the FMCC framework provides us four algorithm versions for clustering: either Algorithm 2.1 or 2.2 for C,F-Coarsening combined with either anti-diagonal or diagonal case for constructing P (and F_c).

Chapter 3

Further Algorithm Details

In the former chapter, we formulated the general FMCC framework, however the clustering problems are usually complicated in practice, e.g., has big noise or unbalanced cluster sizes. In this section, we explain some important details within the formerly described general framework, which can be expected to make the algorithms work better for real problems.

3.1 Parameter Explanations

1. filter threshold θ for strong connections. $\theta \in [0, 1]$ is used to decide the an x-point is strongly connected to a y-point or not. In our experiments, θ is always picked from the interval $[0.5, 0.8]$ according to different data sets. For example, a greater value of θ would be preferable in a people-skill data set, because people typically have many skills, but only those they are very good at can be descriptive on their professions. In practice, one may need to observe several data points before setting the value of θ.

2. Position ratio ρ. $\rho \in [0, 1]$ is used to pick up a C-point according to their connections. $\rho = 1$ means picking the most connected one. We found that $[0.5, 0.8]$ could be a good interval for picking values for ρ in practice. It might be improper to pick a ρ close to 1 because the most connected point might not be representative, for example, the popular words like "the" or "is"; it's also improper to pick a very small ρ because a less connected point will not have enough features to capture the features of the whole group, therefore easily lose group members when picking F-points.

3. Overlap ratio α_x and α_y in Algorithm 2.1. These two parameters are not necessarily equal. In our experiments, we found $[0.4, 0.7]$ is a good interval for those ratios, but one still needs to choose them according to experience and specific situations. For example, in a people-skill data where people are represented by the set X, α_x indicates how many skills two person should overlap if their professions are considered to be the same, while α_y indicates how many common people two skills should overlap if they are considered to be the same kind of profession skills. These knowledge depends on experience as well as our observation on the data.

Finally, we point out that all these three kinds of parameters could influence the speed of the aggregation, as well as the height of the hierarchical clustering tree. For example, a greater θ would induce less strong connections, therefore slower aggregation of those points as well as higher clustering trees; a greater ρ would pick a more connected C-point which labels more F-points in return, therefore increases the aggregation speed.

3.2 Noise Filtering

In this section, we describe an important rule, especially in dealing with the high dimension and big noise data, to filter the noise using a relative threshold value. Let's recall the construction of M_x (or M_y). We first compute $M_x = FP_{y\to x}$ and then normalize each row to sum up to 1. The rationality under this construction is that inner products of the x-points' features describe their proximities. However, if the data is of high dimension and with a lot of noise, these noise could ruin the meaning of the inner products. Take the following special case as an illustrative example. Assume the feature vector of x_i is $F_i = (10, 10, 10, 1, 1, 1, ..., 1) \in R^{1000}$ with 997 1's, and assume two coarse x-points, x_{c1} and x_{c2}, with feature vectors $F_{x_{c1}} = (10, 10, 10, 0, 0, 0, ..., 0) \in R^{1000}$ and $F_{x_{c2}} = (1, 1, 1, 10, 10, 10, 1, ..., 1) \in R^{1000}$. Naturally, i should be a member (at least with high probability) of x_{c1} group in a reasonable classification, because all it's strongly connected features matches x_{c1}'s. However, due to the high dimension noise, i.e., those 1's, the inner product gives $(F_i, F_{cx1}) = 300$ while $(F_i, F_{cx2}) = 1054$. This means the noise dominates the inner products, as a result, makes them meaningless.

To deal with this issue, we apply a filtering procedure to F before computing M_x and M_y. The filtering rule is similar to the construction of strong connections. That is, given a threshold λ, $\forall i, j$, f_{ij} is filtered as zero, if $f_{ij} < \lambda(\max_k f_{kj})$ and $f_{ij} < \lambda(\max_k f_{ik})$. Basically, we intend to filter out those noises and set them as zeros. Note that we will not filter $F_{x\to y}$ or $F_{y\to x}$, because we want to keep all the information of those C-points. After

the filtering, the feature of x_i would become $F_i = (10, 10, 10, 10, 0, 0, ..., 0) \in R^{1000}$, while $F_{x_{c1}}$ and $F_{x_{c2}}$ are kept. Now the inner products reflect their connections more properly: $(F_i, F_{x_{c1}}) = 300$ and $(F_i, F_{x_{c2}}) = 0$.

In practice, the value of λ could be decided by observing the data and noise level.

3.3 Rescaling

Even with filtering procedure, the algorithm could still meet problems, especially on the hierarchical data, when constructing M_x and M_y.

Let's assume the feature vector of x_i as $F_i = (10, 10, 10, 7, 7, 7, 1..., 1) \in R^{1000}$, and assume three coarse x-points, x_{c1}, x_{c2} and x_{c3}, with feature vectors $F_{x_{c1}} = (10, 10, 10, 7, 7, 7, 0, ..., 0) \in R^{1000}$, $F_{x_{c2}} = (7, 7, 7, 10, 10, 10, , ..., 1) \in R^{1000}$ and $F_{x_{c3}} = (1, 1, 1, 1, 1, 1, 10, 10, 10, 1, 1, ..., 1) \in R^{1000}$. This could be a typical part of the hierarchical data in which x_i belongs to x_{c1}'s group in the second level, and the groups of x_{c1} and x_{c2} are combined in the next clustering level.

Now assume we are on the finest (first) level and want to find clustering at the second level. After the filtering procedure in subsection 3.2, we have $(F_i, F_{x_{c1}}) = 447$, $(F_i, F_{x_{c2}}) = 420$ and $(F_i, F_{x_{c3}}) = 0$. The inner products clearly reflect the relation between x_i and x_{c3}, but x_{c1} and x_{c2} almost equally share x_i's membership after normalization.

Since this is the second level, we expect that x_{c1} holds more memberships of x_i. So we reallocate the memberships by applying the exponential rescaling. Specifically, to construct M_x, we rescale the row i of $FP_{y\to x}$ as follows: i. find the maximum and minimum of row i of $FP_{y\to x}$; i.i $(FP_{y\to x})_{ij}$ is rescaled to $(FP_{y\to x})_{ij} exp(\gamma \frac{(FP_{y\to x})_{ij} - min}{max})$, where γ is a parameter to adjust the rescaling scale. Here $(FP_{y\to x})_{ij}$ is multiplied before exponent to keep the zero entries.

By combining subsection 3.2 and 3.3, we describe the actual constructions of M_x and M_y in Algorithm 3.1.

3.4 Normalization

One potential problem in constructing F^c is the unbalanced cluster sizes, since any entry of the coarse feature matrix is the aggregation of its corresponding cluster members therefore bigger clusters will have bigger entry values. In this section, we describe how to normalize the matrix entries by the cluster sizes.

Algorithm 3.1 Construction of M_x and M_y

Input: $F \in R^{m \times n}$, $F_{x \to y} \in R^{m \times n_c}$, $F_{y \to x} \in R^{n \times m_c}$, filtering threshold λ, rescaling scale γ
Output: M_x, M_y

1. Filter F: $\forall i, j$, $f_{ij} = 0$ if $f_{i,j} < \lambda(\max_k\ f_{kj})$ *and* $f_{i,j} < \lambda(\max_k\ f_{i,k})$.
2. $M_x = FP_{y \to x}$; $M_y = F^T P_{x \to y}$.
3. for $i = 1 : m$
   $max \leftarrow \max_j (M_x)_{ij}$; $min \leftarrow \min_j (M_x)_{ij}$;
   for $j = 1 : m_c$
   $(M_x)_{ij} = (M_x)_{ij} exp(\gamma \frac{(M_x)_{ij} - min}{max})$;
   endfor
   endfor
4. for $i = 1 : n$
   $max \leftarrow \max_j (M_y)_{ij}$; $min \leftarrow \min_j (M_y)_{ij}$;
   for $j = 1 : n_c$
   $(M_y)_{ij} = (M_y)_{ij} exp(\gamma \frac{(M_y)_{ij} - min}{max})$;
   endfor
   endfor
5. Linearly normalize each row of M_x and M_y to sum up to 1.

3.4.1 F^c Construction: Anti-Diagonal Case

We firstly consider the anti-diagonal case where $F^c = (P_{y \to x})^T F^T P_{x \to y}$. This direct construction has following drawbacks: firstly, the scale of F^c will grow fast when going to coarse levels; secondly, this construction doesn't have a clear interpretation; thirdly, those more connected C-points will get higher feature intensity values (because they aggregate more points), therefore a higher feature intensity might not result from a stronger connection but a bigger group, i.e., F^c is not descriptive on the clustering property any more.

As a result, the original constructions need some further adjustments. Indeed, instead of $F^T P_{x \to y}$, we use directly M_y (with filtering and rescaling applied). Recall that $P_{y \to x}$

describes all the y-points featured by coarse x-points, so $F^c = (P_{y\to x})^T M_y$ constructs all the coarse y-points featured by coarse x-points by aggregating the memberships of fine y-points captured by M_y. To balance the cluster sizes, we further divide each column by its total memberships of y-points. Formally, the actual F^c is constructed as follows:

$$F^c: \text{ i. } F^c = (P_{y\to x})^T M_y; \text{ ii. Divide the } j\text{'th column of } F^c \text{ by } \sum_{k=1}^{n} (M_y)_{kj} \tag{3.1}$$

Note that this construction is not symmetric–one may also use the form $F^c = (M_x)^T P_{x\to y}$. Experiments show that these two ways do not have obvious difference. One may also use the form $F^c = [(M_x)^T P_{x\to y} + (P_{y\to x})^T M_y]/2$ to make it symmetric. Our experiments on the anti-diagonal case construct F^c using the form in 3.1.

3.4.2 F^c Construction: Diagonal Case

The construction of $F^c = (M_x)^T F M_y$ has proper rationality which is from interpolation. However, similar to the discussion in section 3.4.1, different cluster sizes could still ruin the descriptiveness of the coarse feature matrix. To balance the influence of the cluster size, we also use the following normalization:

$$F^c: \text{ i. } F^c = (M_x)^T F M_y; \text{ ii. Divide } (F^c)_{ij} \text{by } [\sum_{k=1}^{m} (M_x)_{ki}][\sum_{k=1}^{n} (M_y)_{kj}] \tag{3.2}$$

Chapter 4

Complexity

One reason that AMG-like algorithms are efficient is that the problem size becomes smaller and smaller as the grids are coarsened. This makes the cost at the coarsen levels negligible compared with that at the first level. As easily seen, the main computation burden of FMCC lies on the first level. In this chapter, we exam the complexity of FMCC at the first level in details.

We start our analysis from the following simple conclusion which may be considered folklore. It can also be found in [39].

Theorem 9. *([39]) Let $A \in R^{m\times l}$ and $B \in R^{l\times n}$, let c_i denote the number of non-zeros in the i'th column of A while r_i denote the number of non-zeros in the i'th row of B. Then computation of AB can be implemented in time $O(\sum_{i=1}^{l} = c_i r_i)$.*

Note that theorem 9 is not the best lower bound of the complexity of sparse matrix multiplication. For example, [39] presented a sophisticated algorithm for $n \times n$ sparse matrix multiplication with complexity $O(N^{0.7}n^{1.2} + n^{2+o(1)})$ in which N is the number of non-zeros in the sparse matrix. However, the complexity in theorem 9 can be easily implemented and the conclusion is not limited to square matrices. So in our analysis, we use this conclusion for generality sake.

In practice, the original feature intensity matrix F is usually sparse, so we include this property in our analysis. Assume $F \in R^{m\times n}$, r_i and c_j are the numbers of the non-zeros of the i'th row and j'th column of F, respectively. Let $r = \max_{i\le m} r_i$, $c = \max_{j\le n} c_j$ and $N = \sum_{i\le m} r_i = \sum_{j\le n} c_j$ be the number of total non-zeros in F and the coarsened intensity matrix $F^c \in R^{m_c\times n_c}$. The construction of the strong connection matrix S can be implemented in $O(N)$ time simply by comparing those non-zeros. We now analyze the computation complexity for other different parts of FMCC in the following sections.

4.1 Complexity of Alternating C,F-Splitting

In alternating C,F-splitting (Algorithm 2.2): step 1 can be implemented in $O(N+m\log(c)+n\log(r))$ time (using radix sort [12] with complexity $O(m\log(c)+n\log(r))$) which can be further bounded by $O(N)$ by assuming $m\log(c)\leq N$ and $n\log(r)\leq N$ for simplicity; step 2 and 3 visits all the x,y-points and therefore has complexity $O(m+n)$. Since there are m_c coarse x-points and n_c coarse y-points, step 2-4 will be executed for at most $\max(m_c,n_c)$ times, so the the complexity of Algorithm 1 is bounded by $O(\max(m_c,n_c)\times(m+n)+N)$.

4.2 Complexity of Separate C,F-Splitting

Separate C,F-splitting (Algorithm 2.1) has vector dot product in step 3, we first implement SS^T and S^TS beforehand whose complexity is bounded by $O(\sum_i r_i^2+\sum_j c_j^2)=O(N\max(r,c))$ (theorem 9). Then the complexity analysis at each step is as follows: step 1 has complexity $O(N)$; step 2 has complexity $O(m)$; step 3 has complexity $O(m)$ since (S_i,S_{cs}) and $\sum_j S_{ij}$ have been computed in advance. Step 2,3 will be executed for m_c times, so it needs $O(mm_c)$ time to separate x-points, and similarly $O(nn_c)$ time to separate y-points. So the complexity of Algorithm 2 is bounded by $O(N\max(r,c)+mm_c+nn_c)$.

4.3 Complexity of Membership Matrix Constructions

Now we consider the coarse level construction phase. M_x and M_y are constructed in Algorithm 3.1 in which: step 1 has complexity $O(N)$; step 2 has a complexity bounded by $O(\sum_j c_jm_c+\sum_i r_in_c)=O(N\max(m_c,n_c))$ (theorem 9); step 3,4,5 together has a complexity bounded by $O(mm_c+nn_c)$. Since $N\geq\max(m,n)$, Algorithm 3 has complexity $O(N\max(m_c,n_c))$.

4.4 Complexity of F^c Constructions

For generality, we simply assume M_x and M_y are dense matrices. The complexity of the construction of F^c is bounded by $O(\sum_{j=1}^n n_jm_c)=O(Nm_c)$ in the anti-diagonal case . In the diagonal case, there are two ways to compute F^c, i.e., $[(M_x)^TF]M_y$ which has complexity $O(\sum_{i=1}^m m_cr_i)+O(\sum_{j=1}^n m_cn_c)=O(Nm_c+nn_cm_c)$ or $(M_x)^T[FM_y]$ which has complexity

$O(\sum_{j=1}^{n} n_c c_j) + O(\sum_{i=1}^{m} m_c n_c) = O(N n_c + m n_c m_c)$. One can easily choose by comparison a more efficient way of which the complexity could be bounded by $O(N \max(m_c, n_c) + \min(m, n) m_c n_c)$.

4.5 Algorithm Complexities

Now we combine all these complexities. Note that the constructions of M_x and M_y, which has a complexity of $O(N \max(m_c, n_c))$, is common for any version of FMCC. As a result, we conclude the complexity analysis for different FMCC versions as follows:

1. FMCC with Separate C,F-Splitting (Algorithm 2.1) and Anti-diagonal Case (Case 1): $O(N \max(r, c, m_c, n_c))$.
2. FMCC with Alternating C,F-Splitting (Algorithm 2.2) and Anti-diagonal Case (Case 1): $O(N \max(m_c, n_c))$.
3. FMCC with Separate C,F-Splitting (Algorithm 2.1) and Diagonal Case (Case 2): $O(N \max(r, c, m_c, n_c) + m_c n_c \min(m, n))$.
4. FMCC with Separate C,F-Splitting (Algorithm 2.2) and Diagonal Case (Case 2): $O(N \max(m_c, n_c) + m_c n_c \min(m, n))$.

To conclude, Separate C,F-Splitting is at least as expensive as Alternating C,F-Splitting and it becomes more expensive when r or c has big value; the diagonal case is more expensive than the anti-diagonal case because it needs one more matrix multiplication, but it has the same complexity order as the anti-diagonal case unless $m_c n_c \min(m, n)$ has higher order than $N \max(m_c, n_c)$.

In practice, m_c,n_c are usually very small compared with m,n and can be regarded as constants. In such case, the FMCC using Algorithm 2.2 is linear to N and that using Algorithm 2.1 has complexity $O(N \max(r, c))$, which is also proved by our experimental results.

Chapter 5

Experimental Results

In this chapter, we report the experimental results of FMCC as well as the comparisons with other clustering algorithms.

Since FMCC has 4 versions, from now on, we always use "A" and "S" to denote Alternating and Separate C,F-Coarsening, respectively, while "aD" and "D" to denote the anti-diagonal case which constructs F^c using form 2.5 and the diagonal case which constructs F^c using form 2.6 in subsection 2.3.2, respectively. So the FMCC with separate C,F-Coarsening and anti-diagonal case is simply denoted as "FMCC_S&aD. The others are similarly denoted.

5.1 Toy Data

To give the readers an intuitive understanding of a hierarchical data as well as the basics of our algorithms, we start our experiments from a toy data which describes 10 people's skill intensities on 9 different professional skills (the left matrix in Figure 5.1.1). The structure of this data is actually general, we endowed it a specific scenario for a more straightforward understanding.

The data has three clusters of people: Computational Mathematicians (id: 1,2,3), Pure Mathematicians (id: 4,5,6) and Computer Scientists (id: 7,8,9,10). Computational and Pure Mathematicians have some common knowledge on the basic math subjects, e.g., Functional Analysis and Real Analysis, and they are supposed to belong to a bigger group, i.e., Mathematicians. Skills basically have four groups: Computational Math skills (Numerical Analysis, Numerical PDE), Pure Math skills (Abstract Algebra, Algebraic Geometry), basic math skills (Functional Analysis, Real Analysis), and Computer Science skills (Systems,

	Numerical Analysis	Numerical PDE	Systems	Abstract Algebra	Functional Analysis	Networking	Real Analysis	Algebraic geometry	Database
1	10	8	0	1	4	0	4	3	0
2	8	10	0	0	3	0	4	1	3
3	9	9	1	0	5	0	3	0	0
4	4	0	0	9	6	0	6	9	0
5	0	1	0	10	3	1	5	9	0
6	2	4	0	8	5	0	6	10	0
7	2	2	10	0	1	8	1	0	9
8	0	1	8	1	0	10	0	0	10
9	0	1	8	1	0	10	0	0	10
10	1	0	9	0	0	9	2	0	9

	Numerical Analysis	Systems	Functional Analysis	Algebraic Geometry
2	5.66	0.96	4.01	2.52
5	3.28	0.34	4.96	6.66
8	1.37	9.32	0.78	0.32

Level 2

	Numerical Analysis	Systems	Algebraic Geometry
2	1.85	0.36	1.87
8	0.67	3.54	0.37

Level 3

	Numerical Analysis	Systems
2	0.56	0.15
8	0.16	1.5

Level 4

Figure 5.1.1: The feature intensity matrices of the toy data. Left: original feature matrix; Right: coarse feature matrices by FMCC_S&aD

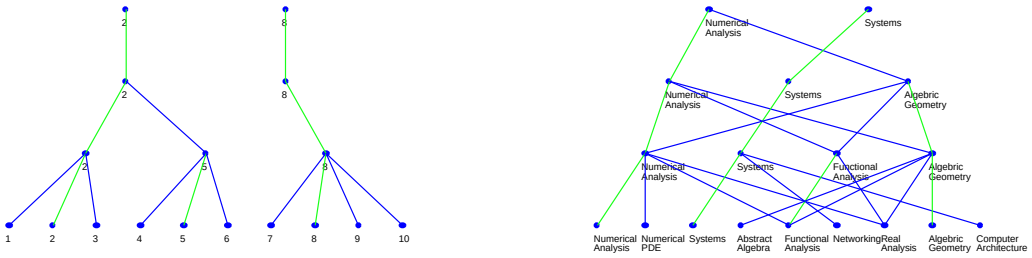

Figure 5.1.2: Left: tree of people with ids; Right: tree of skills. (FMCC_S&aD with filter threshold=0.65)

Networking, Database). Certainly, these math skills are supposed to have some overlap and belong to a bigger group, i.e., the math skill group.

We expect FMCC to find those hierarchical structures. Note that since all the 4 versions of FMCC give similar clustering on the toy data, we only show the results of FMCC_S&aD in this section to avoid redundancy. The systematic comparisons between these 4 versions are left to to the later sections. The right side of Figure 5.1.1 are the coarsened matrices found by FMCC during the execution (row or column labels are the C-point of that group). As we see, the hierarchical structure of the data is clearly uncovered by FMCC, ending at the fourth level with two big groups for both people and skills.

Figure 5.1.2 shows the hierarchical clustering trees. We plot only the strong connections with a filter threshold 0.65 (similar to the construction of S) on the membership matrices M_x and M_y. Green line in the plots means this point becomes a C-point in the next level.

As we see that, the 3 groups of people and 4 groups of skills are found at level 2 and all math skills are combined at level 4.

Figure 5.1.1 and 5.1.2 provide some execution details of FMCC and also show its performance in hierarchical co-clustering. In the next sections, we will compare the performance of the different versions of FMCC with some well-known clustering methods in literature.

5.2 Artificial Hierarchical Data

In this subsection, we design artificial data to systematically test the performance of FMCC and compare with the well-known non-negative matrix factorization (NMF).

Given a feature matrix $F \in R^{m\times n}$, NMF tries to find the *nonnegative* matrices $W \in R^{m\times k}$ and $H \in R^{n\times k}$ which minimizes the difference $||F - WH^T||_2$. Researchers have found that very good co-clustering results can be recovered from the matrix W and H. Basically, the normalization of each row of W (H) will give us the fuzzy clustering of the row (column) points. Here, k corresponds to the number of clusters which has to be specified beforehand. Massive experiments have shown that this factorization provides satisfactory results in many clustering applications, e.g., document classification [38], gene expression data co-clustering [5, 16] and image segmentation [20].

However, NMF has some key limitations. Firstly, NMF is computationally expensive. [37] shows that it's NP-hard to compute the exact NMF, e.g., globally minimize $||F - WH^T||_2$. Some iterative algorithms have been developed to approximate the distance of the difference. Secondly, NMF cannot find the the number of clusters k directly, so it has to be specified beforehand. However, the problem is that it is usually hard to decide a proper k in practice. Thirdly, NMF is a one-level clustering method and cannot find hierarchical clustering structures. In these aspects, FMCC is more "intelligent" than NMF in the sense that it found the whole hierarchical structures automatically. Finally, NMF has a key shortcoming when dealing with co-clustering, that is the number of clusters of row points has to be the same as that of column points. This greatly restricts NMF's function in co-clustering. It meets problems even on the toy data we used in subsection 5.1.

To make the comparison fair, we design the artificial hierarchical data which are particularly suitable for NMF. That is, the artificial data has the same number of row and column clusters. Figure 5.2.1 shows the color maps of an ordered artificial data we constructed and its randomized version, in which deeper color means higher intensity. The artificial data has two cluster levels with 16 small clusters (8 x, y-points in each small block) and

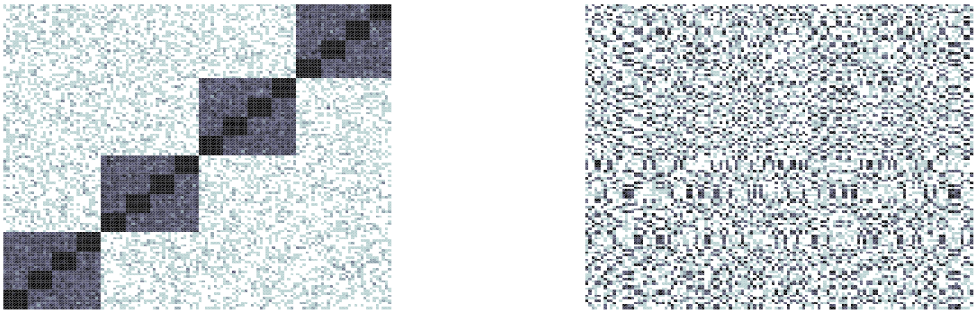

Figure 5.2.1: Color Maps of Feature Matrices. Left: ordered version; Right: randomized version. $\delta = 1$.

4 big ones for either row or column. Every four small clusters belong to a big one. The construction of the feature matrices is described as follows by taking the left map as an example. Let's say the map have three kinds of regions: dark, grey, light (from deep to light), and each region has a "basic intensity": 10 for dark, 5 for grey and 0 for light. Then the intensities in a specific region is randomly chosen as any $i \in \{0, 1, ..., 10\}$ with a probability proportional to the Gaussian distribution form $e^{-\frac{(i-b)^2}{2\sigma^2}}$ where b is the "basic intensity" in that region and the variance σ is a parameter to adjust the noise level. The randomized feature matrix is constructed by randomly permuting all the rows and columns of the ordered one.

We apply FMCC to the above *randomized* matrix in Figure 5.2.1 and try to recover the ordered structure according to the clustering results. Figure 5.2.2 shows the recovered structure by two versions of FMCC. The algorithms almost perfectly recover the hierarchical structures except that both maps have a little flaw in one small block (note that the block order does not influence the comparisons).

In Figure 5.2.3 and 5.2.4, we also show the hierarchical clustering trees of the randomized data in Figure 5.2.1. Both of the two versions end correctly with 3 levels. Figure 5.2.4 shows that FMCC_S&aD correctly found the hierarchical structure: 16 small groups with each four of them enter a big one. However, the FMCC_A&aD in Figure 5.2.3 has some flaws, i.e., 18 small groups for both x, y-points. This result is consistent with our expectation, since alternating C,F-coarsening is not as accurate as separate version. But, in a word, both of the versions give reasonable clustering, though not perfect sometimes.

The above example shows some detailed clustering results given by FMCC, which also experimentally proves its performance. Now, we compare FMCC with NMF systematically on massive artificial data. Since NMF can neither find the number of clusters nor the hier-

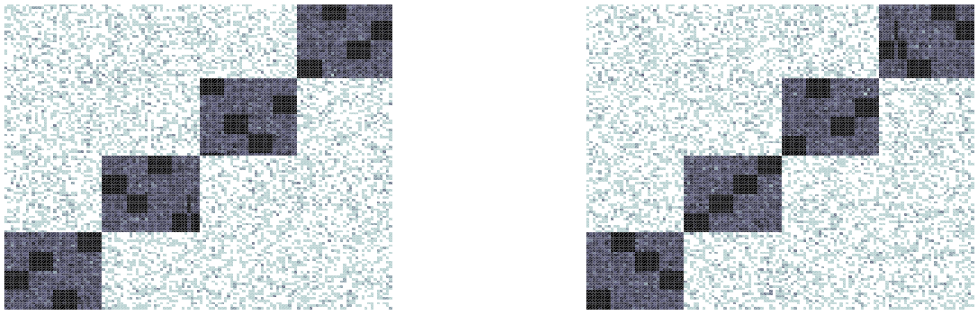

Figure 5.2.2: Color maps recovered by FMCC. Left: FMCC_A&aD; Right: FMCC_S&aD.

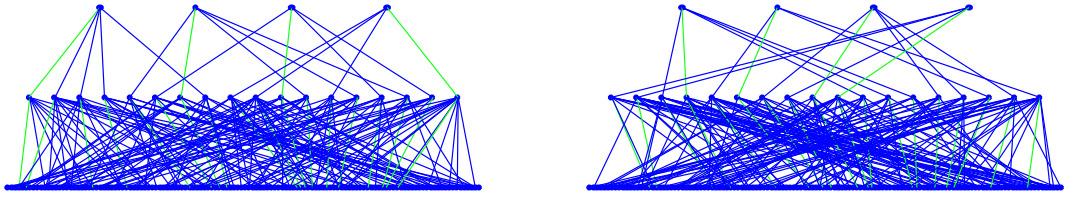

Figure 5.2.3: Hierarchical trees found by FMCC_A&aD. Left: tree of x-points; Right: tree of y-points. filter threshold = 0.8.

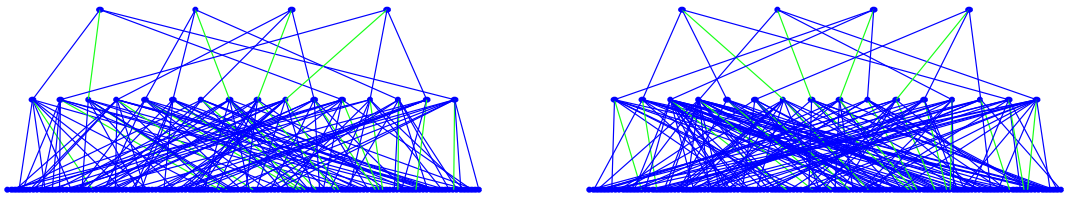

Figure 5.2.4: Hierarchical trees found by FMCC_S&aD. Left: tree of x-points; Right: tree of y-points. filter threshold = 0.8.

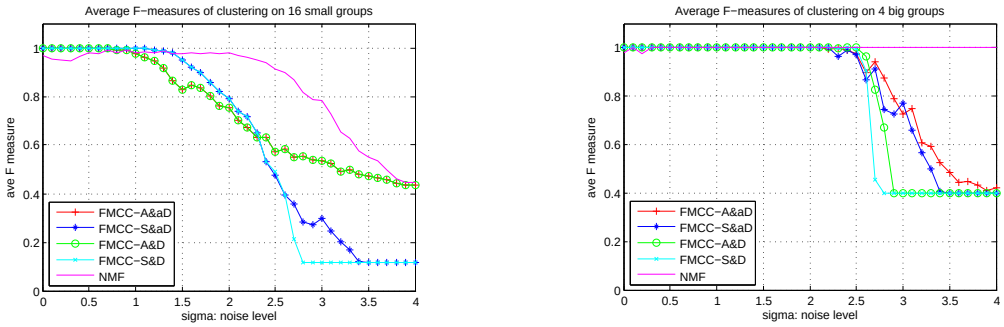

Figure 5.2.5: F-measure Comparison with NMF. Left: small group clustering; Right: big group clustering.

archical structures, we directly pick for NMF the correct number of clusters and compare the clustering results level by level. To quantify the comparison, we use the well-known F-measure ([34], see Appendix A.2.1 for the review). The fuzzy clusters of any algorithm are converted to hard ones by taking the maximum membership of each point.

Figure 5.2.5 shows the performance of FMCC, as well as NMF, at different noise levels. For each noise level, 10 feature matrices (144×144) are randomly generated, and the final F-measure is the average of the F-measures on those 10 matrices and on the clustering of X and Y. We found that FMCC gives better clustering on low noise data, but NMF is more stable on different noise levels. This is reasonable, because we always give NMF the correct numbers of clusters for any noise data, but FMCC has to find them automatically and therefore makes more mistakes as noise grows. The right figure shows that the F^c construction of anti-diagonal case (Case 1) has better performance than the diagonal case (Case 2) on high noise data, while the left figure shows that Separate C,F-Splitting has better performance on low noise data, but becomes worse than Alternating C,F-Splitting when data noise grows bigger. The reason is that higher noise induces more wrong judgments of strong connections, so when the percentage of the wrong judgments increases over some threshold, Alternating C,F-Splitting, which picks C-points by only one strong connection, will have a lower risk to make mistakes than Separate C,F-Splitting which compares all the strong connections.

In these experiments, we intentionally construct the data that are suitable for NMF. We note that it is actually easy for FMCC to beat NMF on the data in which number of row clusters are different from number of column clusters. In Figure 5.2.6, we present the color map (left) of a feature matrix which has 3 row clusters and 6 column clusters. When

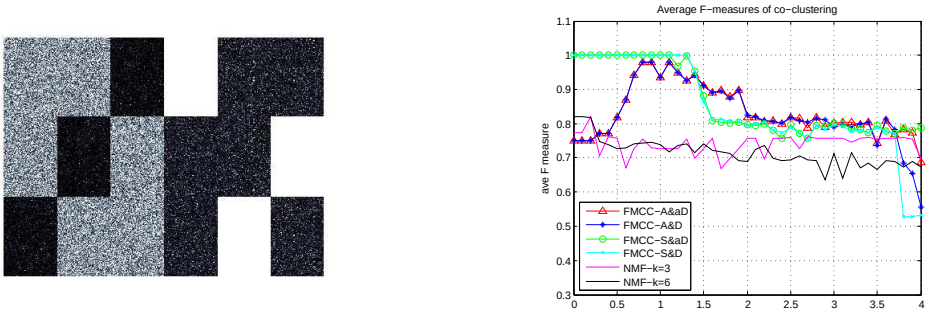

Figure 5.2.6: Left: ranked feature matrix (384 × 384,σ=3); Middle: average F-measures on different noise levels.

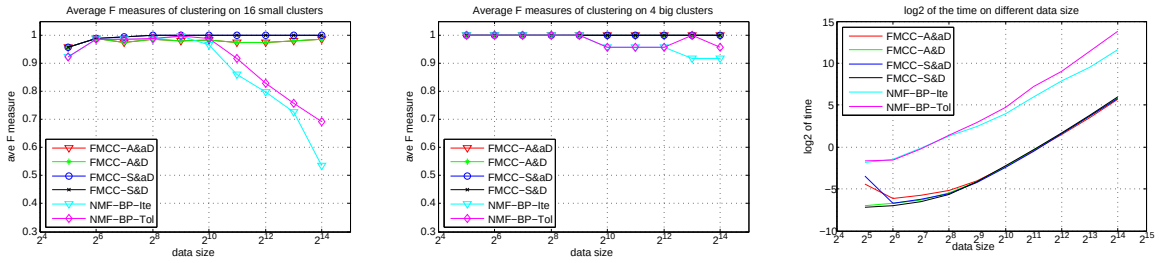

Figure 5.2.7: Performance on different data size. Left: average F-measure of small cluster level; Middle: average F-measure of big cluster level; Right: time performance.

FMCC and NMF are applied on such kind of data (still, rows and columns are randomly permuted), FMCC has better performance than NMF almost on every noise level.

We further test the stability and scalability of these algorithms on data size. We generate similar data structure as Figure 5.2.1 with $\sigma = 1$, but increase the number of points in each small cluster. To test the time performance, we use the algorithm developed in [17] in 2008 which is based on alternating non-negativity constrained least squares and block principal pivoting (NMF-BP), and is one of the fastest NMF iterative algorithms in current literature. The MATLAB code of NMF-BP with recommended parameters is downloaded from the authors' online link. FMCC is also implemented in MATLAB. The processor we used is 2.5 GHz Intel Core i5 with 16GB 1333 MHz DDR3 memory on OS X 10.8.3 (12D78).

In the experiments we found that NMF-BP needs more iterations to converge when data size grows, so we use NMF-BP with two different stop criterion: iterate 100 times *or* the

stopping tolerance (10^{-3}) is achieved (NMF-BP-Ite) and iterate enough *until* the stopping tolerance is achieved (NMF-BP-Tol). Figure 5.2.7 shows the performance of these algorithms. We found that FMCC is stable on data size and the time performance also shows experimentally that complexity of FMCC is linear to the data size N (in this data m_c and n_c are small). However, the clustering performance of NMF decreases as data size grows even when the stopping tolerance is achieved (NMF-BP-Tol). The log-time plot of NMF-BP-Ite is almost parallel to FMCC asymptotically, this is because each iteration of NMF-BP has the same complexity as FMCC (basically matrix multiplications), however NMF-BP needs many iterations to converge. To sum up, FMCC beats NMF-BP in both the clustering and time performance on the artificial data of large size.

5.3 Gene Expression Data

Bioinformatics and text data mining are two popular fields for co-clustering. In this section, we compare FMCC with some well-known co-clustering algorithms in bioinformatics for gene expression data.

[28] provides a systematic comparison of 7 co-clustering methods for gene expression data. Those methods are: their own method Bimax [28]; Cheng and Church's algorithm CC [6]; Samba [36]; Order Preserving Submatrix Algorithm, OPSM [2]; Iterative Signature Algorithm, ISA [13]; xMotif [24]; the standard hierarchical clustering method (HCL) in MATLAB.

In the comparison, we use directly the Scenario 1 data and their clustering results of all these methods (except FMCC) provided on the online supporting page of the paper [28]. Following [28], we also use the *gene match score* as the comparison measure, which is defined as follows:

Definition 10. ([28]) Let M_1, M_2 be two sets of biclusters. The gene match score of M_1 with respect to M_2 is given by the function

$$S_G^*(M_1, M_2) = \frac{1}{|M_1|} \sum_{(G_1,C_1)\in M_1} \max_{(G_2,C_2)\in M_2} \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|}$$

where G and C correspond to gene cluster and condition cluster, respectively.

Figure 5.3.1 shows the comparison of all the algorithms on different noise levels, while Figure 5.3.2 shows the comparison on different overlap levels, i.e., clusters have overlapped

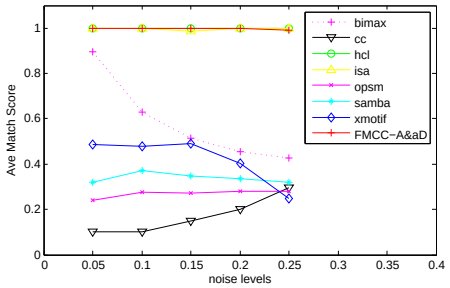

Figure 5.3.1: average gene match score on different noise level.

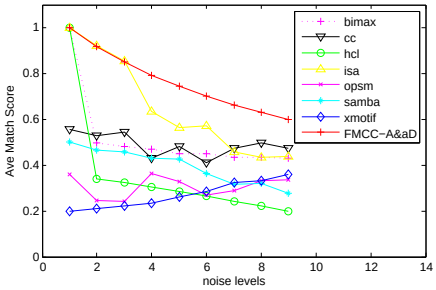

Figure 5.3.2: average gene match score on different overlap level.

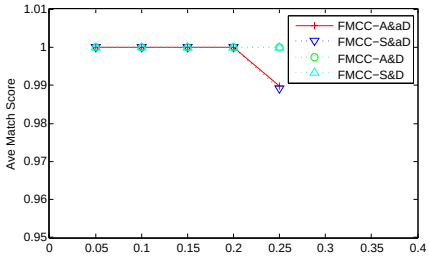

Figure 5.3.3: Different Versions of FMCC on Different Noise Levels.

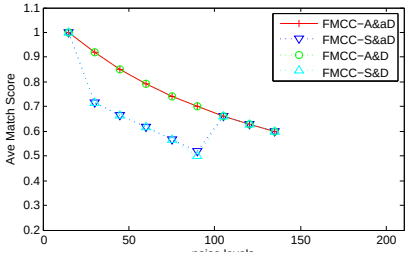

Figure 5.3.4: Different Versions of FMCC on Different Overlap Levels.

members. On the noise data, FMCC almost gives perfect clustering like hcl and isa which, however, have much worse performance than FMCC on the overlap data. These results show that in the gene data set, FMCC is competitive even to the best of those tested biclustering algorithms which are specifically designed for gene expression data.

We also compare different versions of FMCC on the Gene Expression Data in Figure 5.3.3 and Figure 5.3.4. These figures show that the F^c construction of diagonal case achieves better performance than that of anti-diagonal case on the noise data, while Alternating C,F-Splitting achieves better performance than separate version on the overlap data.

5.4 Actual LinkedIn Data

We further apply FMCC to an actual LinkedIn data set and report the experimental results in this section. Our data set is a 176×47 matrix in which each row corresponds to

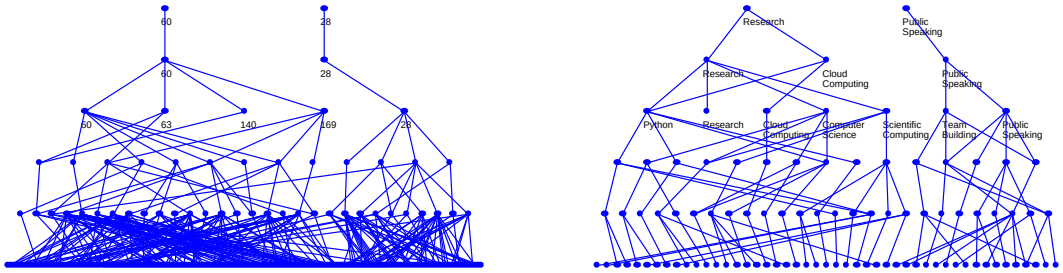

Figure 5.4.1: Hierarchical trees found by FMCC_A&aD. Left: tree of users; Right: tree of skills. filter threshold = 0.8

a LinkedIn user and each column corresponds to a skill listed on the homepage of the user. This data set is suitable for hierarchical co-clustering, because both the LinkedIn users and skills can be hierarchically classified according to their professions (see also section 5.1 toy data).

The data is originally harvested by starting from randomly picked 3 seed users (in the fields of Numerical Analysis, Physics and Computer Science, respectively) and then doing breadth first search of all their endorsers until we can not visit an endorser's homepage (note that a common LinkedIn user can only visit the other users who are within 3 connections to him). We collected all the users we have visited which consists of 1247 users and 7074 skills. However, many skills show up very rare or overlap with each other, and some users have very few skills. To get a more compact data, we do the following filtering: firstly, the skills are filtered if less than 70 of those 1247 users know them; then, the user are filtered if they have less than 7 skills after the skill filtering phase. After the filtering, we get the final 176×47 data set.

Former experiments have shown that the clustering performance of the 4 versions of FMCC does not have obvious difference, we only present the results of FMCC_S&aD in this section to avoid redundancy. The algorithm ends with 6 levels in which level 4,5,6 show meaningful community interpretation. Figure 5.4.1 shows the trees of users and skills found by FMCC_S&aD, in which we list the user ids and skill names of the coarse points at the top 3 levels. One may found that both of the trees seem to have a gap and be divided into two parts. This is because we re-ordered the original users and skills according to the clustering results of NMF with $k = 2$ by putting users (or skills) within one cluster to one side when drawing the trees (but FMCC_S&aD is applied to the original data without re-ordering). We can see from these two trees that the two clusters at the finest level are almost the same as those found by NMF. However, the interesting observation is

that the skill tree is more clearly separated into two parts than the user tree, one possible explanation is that skills have clearer clustering structures than users, because people can affiliate to different professions, which makes them simultaneously belong to different profession clusters.

We further plot the skill memberships of the coarse skill points at different levels (Figure 5.4.2). Those plots show that the top memberships of the skill clusters differ from each other and are well-separated. To show the details, we list the top 3 memberships of each skill clusters in Table 5.1 which also interprets the clusters and hierarchical structures.

Figure 5.4.3 shows the user memberships of the coarse user points at different levels. They are also well-separated. Since, the actual professions of the users are not known to us, the user clusters cannot be interpreted simply by looking at their ids. Instead, we aggregate the features of the user clusters according to their memberships. Table 5.2 lists the top features of different user groups. We found that the user clusters at level 4 share some common top features, because some features, e.g., Computer Science or Algorithms, are so popular that most users have such skills whichever group they are in. We then turn to look at the top "differentiating features[1]" which give us clear community interpretation of different user clusters. Note that user clusters keep the same at level 5 and level 6, but skill groups are merged from level 5 to 6.

[1]A skill is differentiating to a user cluster if it holds the greatest intensity in this user cluster and the intensity difference between the first and second greatest is bigger than the intensity difference between the second and third greatest.

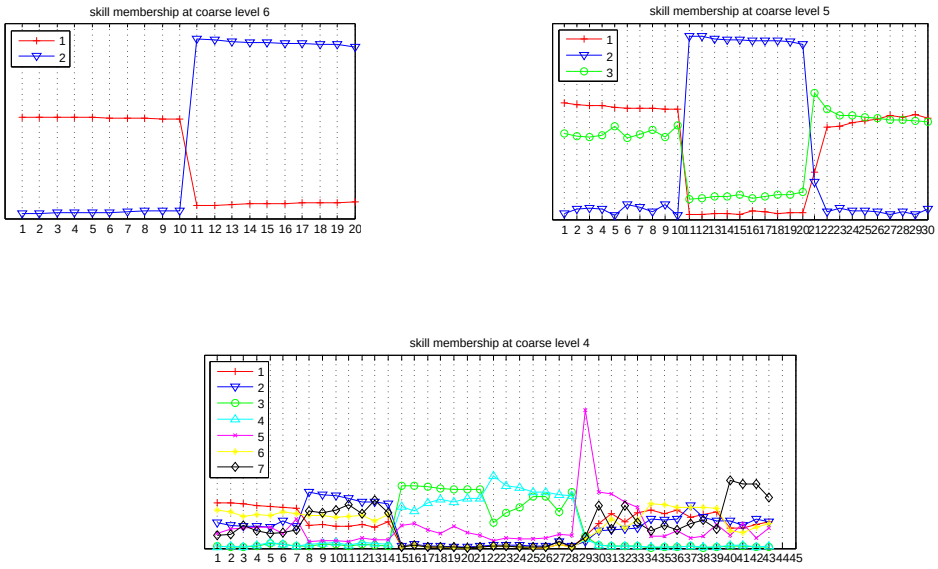

Figure 5.4.2: Skill Memberships at different levels

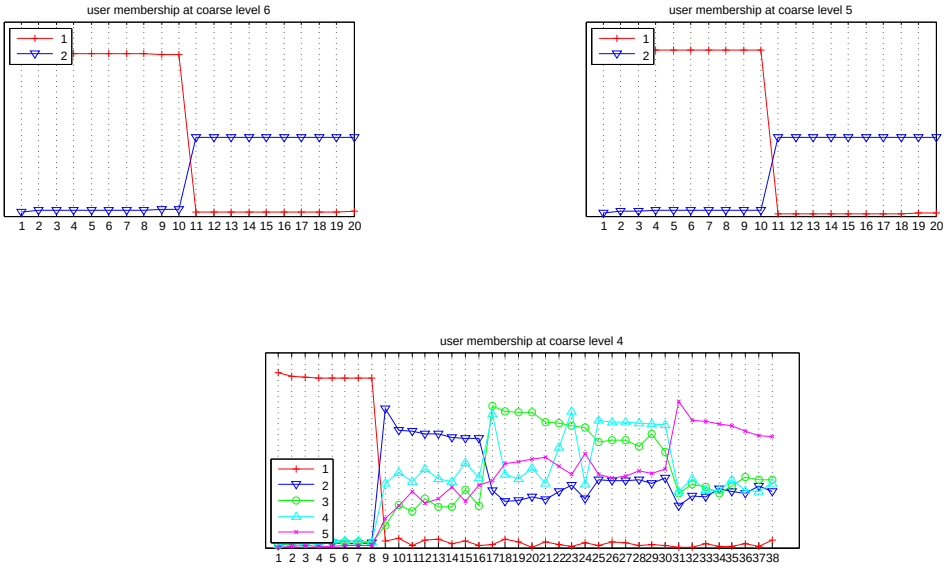

Figure 5.4.3: User Memberships at different levels

level	nodeID	Top Memberships	Interpretations	n_member
6	1	Databases Latex Matlab	Techniques	31.2
6	2	Management Consulting Change Management Social Media	Business	15.8
5	1	Mathematical Modeling Data Mining Numerical Analysis	Mathematics	16.1
5	2	Management Consulting Change Management Social Media	Business	15.7
5	3	Cloud Computing Distributed Systems HPC	Computer Science	15.2
4	1	Java Software Engineering Python	Software Engineering	9
4	2	Data Mining Business Intelligence Data Analysis	Data Mining	2.6
4	3	Business Development Start-ups Team Building	Entrepreneur	6.6
4	4	Public Speaking Project Management Business Analysis	Management and Media	10
4	5	Cloud Computing HPC Distributed Systems	High Performance Computing	3.7
4	6	Databases Matlab Programming	Industrial Data Science Skills	8.1
4	7	Numerical Analysis Scientific Computing Mathematical Modeling	Numerical Skills	7

Table 5.1: the top memberships and interpretations of coarse skill groups at different levels

level	nodeID	Top Features	Differentiating Features	Interpretation	n_member
6	1	Business Strategy Strategic Planning Entrepreneurship	Business Strategy Strategic Planning Entrepreneurship	Business People	55.7
6	2	Algorithms Computer Science HPC	Algorithms Computer Science HPC	Technical People	120.3
5	1	Business Strategy Strategic Planning Entrepreneurship	Business Strategy Strategic Planning Entrepreneurship	Business People	54.4
5	2	Algorithms Computer Science HPC	Algorithms Computer Science HPC	Technical People	121.6
4	1	Business Strategy Strategic Planning Entrepreneurship	Business Strategy Strategic Planning Entrepreneurship	Business People	54.8
4	2	HPC Algorithms Software Development	Software Development Computer Science C++	Software Developer	52.8
4	3	Algorithms Machine Learning Scientific Computing	Machine Learning Data Mining Research	ML and Data Mining People	22.4
4	4	Scientific Computing HPC Algorithms	Scientific Computing HPC Numerical Analysis	HPC People	4.8
4	5	Algorithms Computer Science Machine Learning	Algorithms Java Matlab	Data Scientists	41.2

Table 5.2: the features and interpretations of coarse user groups at different levels

Chapter 6

Conclusion

In this paper, we presented a AMG-inspired multilevel framework for hierarchical co-clustering. The fast multilevel co-clustering (FMCC) framework implements a bi-coarsening process recursively on the bipartite graph induced by the feature matrix, producing a hierarchy of overlapping co-clusters and their connections. Within the general framework, two C,F-splitting algorithmes and two fashions of coarse feature matrix construction are proposed, respectively. Compared with other co-clustering algorithms, FMCC algorithms have the following advantages: they are computationally efficient (almost linear in the data size); there is no need to specify the number of clusters since FMCC finds it automatically; and the clustering gives hierarchical structure for both the row and the column variables. FMCC algorithms produce interpretable co-clusters on several recursive levels along with information on how they are connected. Those algorithms are accurate, fast and scalable, as demonstrated by numerical tests on co-clustering problems with synthetic and real data from the fields of gene expression data and online social networks.

APPENDICES

Appendix A

A Review of Clustering Measures

In this section, we give a brief literature review about the measures to compare different clustering results. We found that there have not been widely-used co-clustering measures in literature, so all these reviews are simply for clustering.

A.1 Weighted Graph Measures

Let $A \in \mathbb{R}^{n \times n}$ be the weighted adjacency matrix of a graph $G(V, E)$. The element of A is denoted by $a_{i,j}$. For example, A can be obtained from the $x - y$ feature matrix F by $A = FF^T$. This subsection reviews the graph clustering measures that directly compare the clusters with the original graph information, without known clustering groundtruth.

A.1.1 Measures for one Single Hard Cluster

Let node subset $C \subseteq V$ be a single cluster. We are interested in the prominence of the cluster C. For simplicity, let $\bar{C}$ denote the complementary set $V \setminus C$, i.e., $C \cup \bar{C} = V$ and $C \cap \bar{C} = \emptyset$.

For a better interpretation, let $i_c = \sum_{i \in C, j \in C} a_{i,j}$ be the sum of the interior weights within cluster C, $b_c = \sum_{i \in C, j \in \bar{C}} a_{i,j}$ be the sum of the weights between cluster C and $\bar{C}$.

- Modularity

Modularity is originally proposed for unweighted graph ([25]), however the definition can be naturally generalized for a weighted one.

For any $i \in V$, let d_i be the weighted degree of node i, i.e., $d_i = \sum_{j \in V} a_{ij}$. Let m be the sum of total weighted edges, so $\sum_{i \in V} d_i = 2m$. The modularity of weighted graph is defined as follows:

$$\phi(C) = \frac{1}{2m} \sum_{i \in C, j \in C} (a_{i,j} - \frac{d_i d_j}{2m})$$

A bigger modularity $\phi(C)$ means a more community-like cluster C. A very loose bound for $\phi(C)$ is $[-1, 1]$.

- Conductance [15]

$$\phi(C) = \frac{\sum\limits_{i \in C, j \in \bar{C}} a_{i,j}}{\min(\sum\limits_{i \in C, j \in V} a_{i,j}, \sum\limits_{i \in \bar{C}, j \in V} a_{i,j})} = \frac{\sum\limits_{i \in C, j \in \bar{C}} a_{i,j}}{\sum\limits_{i \in C, j \in \bar{C}} a_{i,j} + \min(\sum\limits_{i \in C, j \in C} a_{i,j}, \sum\limits_{i \in \bar{C}, j \in \bar{C}} a_{i,j})} = \frac{b_c}{b_c + \min(i_c, i_{\bar{c}})}$$

A smaller conductance $\phi(C)$ means a more community-like cluster C. $\phi(C) \in [0, 1]$ and $\phi(C) = \phi(\bar{C})$. $\phi(C) = 0$ means no edges between C and $\bar{C}$, while $\phi(C) = 1$ means C or $\bar{C}$ has no edges inside.

- Normalized Cut [32]

$$\phi(C) = \frac{\sum\limits_{i \in C, j \in \bar{C}} a_{ij}}{\sum\limits_{i \in C, j \in V} a_{ij}} + \frac{\sum\limits_{i \in C, j \in \bar{C}} a_{ij}}{\sum\limits_{i \in \bar{C}, j \in V} a_{ij}} = \frac{b_c}{i_c + b_c} + \frac{b_c}{i_{\bar{c}} + b_c}$$

A smaller conductance $\phi(C)$ means a more community-like cluster C. $\phi(C) \in [0, 2]$ and $\phi(C) = \phi(\bar{C})$. $\phi(C) = 0$ means no edges between C and $\bar{C}$, while $\phi(C) = 2$ means both C and $\bar{C}$ have no edges inside.

- Saliency

[18] proposed the following saliency measure:

$$\phi(C) = \frac{\sum\limits_{i \in C, j \in \bar{C}} a_{ij}}{\sum\limits_{i \in C, j \in C} a_{ij}} = \frac{b_c}{i_c}$$

Normalized Saliency is defined as follows ([14]):

$$\phi_{norm}(C) = \frac{(\sum_{i\in C, j\in \bar{C}} a_{ij})/(\sum_{i\in C, j\in \bar{C}} sgn(a_{ij}))}{(\sum_{i\in C, j\in C} a_{ij})/(\sum_{i\in C, j\in C} sgn(a_{ij}))}$$

where $sgn(x) = 0$ when $x = 0$, otherwise $sgn(x) = 1$.

A smaller saliency (or normalized) means a more community-like cluster C.

A.1.2 Generalization to Multiple Clusters

The above cluster measures can be naturally generalized to the case of multiple clusters. Given $G(V, E)$, now assume we have K hard clusters $\mathcal{C} = C_1, ..., C_K$ with $C_i \cap C_j = \emptyset, \forall i \neq j$ and $\cup_i C_i = V$.

- Modularity:

$$\phi(\mathcal{C}) = \frac{1}{2mK} \sum_{k=1}^{K} \sum_{i,j\in C_k} (a_{i,j} - \frac{d_i d_j}{2m})$$

where $d_i = \sum_{j\in V} a_{ij}$ and $\sum_{i\in S} d_i = 2m$.

- Conductance:

$$\phi(\mathcal{C}) = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{i\in C_k, j\in \bar{C_k}} a_{i,j}}{\sum_{i\in C_k, j\in V} a_{i,j}}$$

- Normalized Cut:

$$\phi(\mathcal{C}) = \frac{1}{K} \sum_{k=1}^{K} (\frac{\sum_{i\in C_k, j\in \bar{C_k}} a_{ij}}{\sum_{i\in C_k, j\in V} a_{ij}} + \frac{\sum_{i\in C_k, j\in \bar{C_k}} a_{ij}}{\sum_{i\in \bar{C_k}, j\in V} a_{ij}})$$

- Saliency:

$$\phi(\mathcal{C}) = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{i\in C_k, j\in \bar{C_k}} a_{ij}}{\sum_{i\in C_k, j\in C_k} a_{ij}}$$

A.1.3 Generalization to Soft Clusters

Now we generalize the clustering measures to soft clusters. Given $G(V, E)$, assume we have K soft clusters $\mathcal{C} = C_1, ..., C_K$. For any $i \in V$, $p_{i,c_k}$is the probability that i is in cluster C_k.

- Generalized Modularity

$$\phi(\mathcal{C}) = \frac{1}{2mK} \sum_{k=1}^{K} \sum_{i \in C_k, j \in C_k} (p_{i,c_k} p_{j,c_k} a_{i,j} - \frac{d_i d_j}{2m})$$

where $d_i = \sum_{j \in V} a_{ij}$ and $\sum_{i \in S} d_i = 2m$.

Note that the above definition will degenerate naturally to the hard clustering case when $p_{i,c_k} \in \{0, 1\}$.

An interesting observation is that the soft clustering problem using $\phi(\mathcal{C})$ can be formulated as an optimization problem as follows:

$$\max_{p_{i,c_k}, K} \frac{1}{4mK} \sum_{k=1}^{K} \sum_{i,j=1}^{|V|} (p_{i,c_k} p_{j,c_k} a_{i,j} - \frac{d_i d_j}{2m})$$
$$s.t \sum_{k=1}^{K} p_{i,c_k} = 1, \ \forall i$$
$$p_{i,c_k} \geq 0$$

Recalling that the hard community detection using modularity is NP-hard due to the discrete nature of hard clustering. The above formulation can be regarded as a relaxation of the original hard clustering. This relaxed optimization problem can be solved efficiently.

This potentially could be an interesting topic.

- Generalized Conductance

The generalized conductance could be defined as following:

$$\phi(\mathcal{C}) = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{i,j=1}^{|V|} p_{i,c_k}(1 - p_{j,c_k}) a_{i,j}}{\sum_{i,j=1}^{|V|} p_{i,c_k} a_{i,j}}$$

Again, this is a relaxation of the original conductance definition. When it's a hard clustering, it degenerate to the original conductance definition. It can also be formulated as an optimization problem, too.

- Generalized Normalized Cut:

$$\phi(\mathcal{C}) = \frac{1}{K}\sum_{k=1}^{K}(\frac{\sum_{i,j=1}^{|V|} p_{i,c_k}(1-p_{j,c_k})a_{i,j}}{\sum_{i,j=1}^{|V|} p_{i,c_k}a_{i,j}} + \frac{\sum_{i,j=1}^{|V|} p_{i,c_k}(1-p_{j,c_k})a_{i,j}}{\sum_{i,j=1}^{|V|} (1-p_{i,c_k})a_{i,j}})$$

- Generalized Saliency:

$$\phi(\mathcal{C}) = \frac{1}{K}\sum_{k=1}^{K}\frac{\sum_{i,j=1}^{|V|} p_{i,c_k}(1-p_{j,c_k})a_{i,j}}{\sum_{i,j=1}^{|V|} p_{i,c_k}p_{j,c_k}a_{i,j}}$$

A.2 Cluster Measures

In this subsection, we review the measures which measure the difference between two clustering. These measures usually are used with groundtruth (or well-clustered results) to compare the difference between a clustering and the groundtruth.

A.2.1 Hard Cluster Measures

Let $\mathcal{D}$ be the data set of n points, $\mathcal{C} = C_1, ..., C_{K_0}$ is a hard clustering of $\mathcal{D}$, i.e., $C_i \cap C_j = \emptyset, \forall i \neq j$ and $\cup_i C_i = \mathcal{D}$. Let $\mathcal{C}' = C'_1, ..., C'_{K_1}$ be another clustering of $\mathcal{D}$. We are interested in the measures to compare the difference between $\mathcal{C}$ and $\mathcal{C}'$. Let n_k be the number of elements in cluster C_k, while n'_k is the number of elements in cluster C'_k.

- Confusion Matrix

Following matrix $N \in R^{K_0 \times K_1}$ is the so-called confusion matrix:

$$n_{k_0 k_1} = |C_{k_0} \cap C'_{k_1}|$$

Confusion matrix is the most direct way to compare two clustering. But the disadvantage is that there are no standards for the comparison, i.e., no quantitative measure values. This is why the following standard measures are proposed. In a word, they are used for standardizing human's judgements.

- Matching Sets

Let π be any map, mapping a cluster $C_{k_0} \in \mathcal{C}$ to a cluster $C'_{k_1} \in \mathcal{C}'$. Meila and Heckerman ([23]) computed the criterion which is the proportion of the shared elements between $\mathcal{C}$ and $\mathcal{C}'$:

$$\phi(\mathcal{C}, \mathcal{C}') = 1 - \frac{1}{n} \max_{\pi} \sum_{k} n_{k,\pi(k)}$$

in which "$\max_{\pi}$" chooses the best mapping.

The smaller $\phi(\mathcal{C}, \mathcal{C}')$ is, the closer $\mathcal{C}$ and $\mathcal{C}'$ are. This measure is not symmetric. A zero distance means $\mathcal{C}$ completely determines $\mathcal{C}'$, or $\mathcal{C}'$ is determined by merging some clusters of $\mathcal{C}$.

- Mutual Information

Firstly, we define entropy of a clustering $\mathcal{C}$. Given any point $i \in \mathcal{D}$, the probability of i being in cluster C_k is n_k/n. So the entropy is defined as following:

$$\mathcal{H}(\mathcal{C}) = -\sum_{k} \frac{n_k}{n} log \frac{n_k}{n}$$

Naturally, the mutual information between two clusters $\mathcal{C}$ and $\mathcal{C}'$ is defined as following:

$$\mathcal{I}(\mathcal{C}, \mathcal{C}') = \sum_{k_0} \sum_{k_1} \frac{n_{k_0 k_1}}{n} log \frac{n_{k_0 k_1}/n}{(n_{k_0}/n)(n'_{k_1}/n)}$$

Intuitively, entropy describes the information a clustering has, while mutual information describe the common information two clustering share. Here are some properties of mutual information:

1. non-negativity and symmetric:

$$\mathcal{I}(\mathcal{C}, \mathcal{C}') = \mathcal{I}(\mathcal{C}', \mathcal{C}) \geq 0$$

2. a more intuitive representation:

$$\mathcal{I}(\mathcal{C},\mathcal{C}') = \mathcal{H}(\mathcal{C}) + \mathcal{H}(\mathcal{C}') - \mathcal{H}(\mathcal{C},\mathcal{C}')$$

3. mutual information is less than entropy:

$$\mathcal{I}(\mathcal{C},\mathcal{C}') \leq \min(\mathcal{H}(\mathcal{C}), \mathcal{H}(\mathcal{C}'))$$

   Equality occurs when one clustering completely determines the other. For example, if $\mathcal{C}'$ is obtained from $\mathcal{C}$ by merging two or more clusters.

Normalized mutual information can be defined by either of the following two ways [21]:

$$\mathcal{I}_{norm}(\mathcal{C},\mathcal{C}') = \frac{\mathcal{I}(\mathcal{C},\mathcal{C}')}{\max(\mathcal{H}(\mathcal{C}), \mathcal{H}(\mathcal{C}'))}$$

or ([7])

$$\mathcal{I}_{norm}(\mathcal{C},\mathcal{C}') = \frac{2\mathcal{I}(\mathcal{C},\mathcal{C}')}{\mathcal{H}(\mathcal{C}) + \mathcal{H}(\mathcal{C}')}$$

- Variation of Information [22]

Variation of information (VI) is defined as following:

$$VI(\mathcal{C},\mathcal{C}') = \mathcal{H}(\mathcal{C}) + \mathcal{H}(\mathcal{C}') - 2\mathcal{I}(\mathcal{C},\mathcal{C}') = \mathcal{H}(\mathcal{C}|\mathcal{C}') + \mathcal{H}(\mathcal{C}'|\mathcal{C})$$

Intuitively, VI is the information which is not shared by $\mathcal{C}$ and $\mathcal{C}'$. Here are some properties of VI:

1. VI is a metric between $\mathcal{C}$ and $\mathcal{C}'$, satisfying symmetry, non-negativity and $VI(\mathcal{C},\mathcal{C}') = 0$ if and only if $\mathcal{C} = \mathcal{C}'$, and the following triangle inequality:

$$VI(\mathcal{C}_1,\mathcal{C}_2) + VI(\mathcal{C}_2,\mathcal{C}_3) \geq VI(\mathcal{C}_1,\mathcal{C}_3)$$

2. the upper bound: $VI(\mathcal{C},\mathcal{C}') \leq \log n$

3. if $\mathcal{C}'$ is obtained by splitting $C_k \in \mathcal{C}$ (with n_k points) into $C_{k1}, ..., C_{kl}$, then:

$$VI(\mathcal{C},\mathcal{C}') = \frac{n_k}{n}(\sum_{m=1}^{l} \frac{n_{km}}{n_k} log \frac{n_{km}}{n_k})$$

The normalized VI is given by following form ([21]):

$$VI_{norm}(\mathcal{C}, \mathcal{C}') = \frac{\mathcal{H}(\mathcal{C}|\mathcal{C}')}{\mathcal{H}(\mathcal{C})} + \frac{\mathcal{H}(\mathcal{C}'|\mathcal{C})}{\mathcal{H}(\mathcal{C}')}$$

- F-measure ([34])

F-measure is defined based on recall and precision. Specifically, assume $\mathcal{C}$ be the groundtruth class and $\mathcal{C}'$ be any cluster. For any cluster $C'_{k_1} \in \mathcal{C}'$ and class $C_{k_0} \in \mathcal{C}$:

$$Recall(C_{k_0}, C'_{k_1}) = n_{k_0 k_1}/n_{k_0}$$

$$Precision(C_{k_0}, C'_{k_1}) = n_{k_0 k_1}/n'_{k_1}$$

The F-measure of cluster C_{k_0} and class C'_{k_1} is given by:

$$F(C_{k_0}, C'_{k_1}) = \frac{2 * Recall(C_{k_0}, C'_{k_1}) * Precision(C_{k_0}, C'_{k_1})}{Recall(C_{k_0}, C'_{k_1}) + Precision(C_{k_0}, C'_{k_1})} = \frac{2n_{k_0 k_1}}{n_{k_0} + n'_{k_1}}$$

Note that $F(C_{k_0}, C'_{k_1})$ is symmetric. Then F-measure of cluster $\mathcal{C}'$ and class $\mathcal{C}$ is given by the following weighted sum:

$$F(\mathcal{C}, \mathcal{C}') = \sum_{k_0} \frac{n_{k_0}}{n} \max_{k_1} \{F(C_{k_0}, C'_{k_1})\}$$

$F(\mathcal{C}, \mathcal{C}') \in [0, 1]$. $F(\mathcal{C}, \mathcal{C}') = 1$ if and only if $\mathcal{C}$ and $\mathcal{C}'$ are exactly the same. Note that F measure is not symmetric, that is $F(\mathcal{C}', \mathcal{C}) \neq F(\mathcal{C}, \mathcal{C}')$ generally. Usually, $\mathcal{C}$ is used as a groundtruth.

A.2.2 Generalization to Soft Clustering

Now we generalize $\mathcal{C} = C_1, ..., C_{K_0}$ and $\mathcal{C}' = C'_1, ..., C'_{K_1}$ to soft clustering case. For any $i \in V$, let $p_{i,c_{k_0}}$ and $p_{i,c'_{k'_1}}$ be the probabilities that i is in C_{k_0} and C'_{k_1}, respectively. Let $n_{k_0} = \sum_{i=1}^{|V|} p_{i,c_{k_0}}$ and $n'_{k_1} = \sum_{i=1}^{|V|} p_{i,c'_{k_1}}$. $|V| = n$.

- Generalized Confusion Matrix

Following matrix $N \in R^{K_0 \times K_1}$ is the generalized confusion matrix:

$$n_{k_0 k_1} = \sum_{i=1}^{n} p_{i,c_{k_0}} p_{i,c'_{k_1}}$$

where $n_{k_0 k_1}$ denotes the number of elements that C_{k_0} and C'_{k_1} share.

This degenerates to the original definition when $p \in \{0, 1\}$. Note that we still have $\sum_{k_0,k_1} n_{k_0 k_1} = n$.

- Generalized Matching Sets

Let π be any map, mapping $C_{k_0} \in \mathcal{C}$ to $C'_{k'_1} \in \mathcal{C}'$. The following is a generalized set matching measure:

$$\phi(\mathcal{C}, \mathcal{C}') = 1 - \frac{1}{n} \max_{\pi} \sum_{k} n_{k,\pi(k)}$$

- Generalized Mutual Information and VI

Entropy and mutual information have exactly the same forms as the original one, in which n_k and $n_{k_0 k_1}$ use the generalized expressions. Entropy:

$$\mathcal{H}(\mathcal{C}) = -\sum_{k} \frac{n_k}{n} log \frac{n_k}{n}$$

The mutual information between two clusters $\mathcal{C}$ and $\mathcal{C}'$:

$$\mathcal{I}(\mathcal{C}, \mathcal{C}') = \sum_{k_0} \sum_{k_1} \frac{n_{k_0 k_1}}{n} log \frac{n_{k_0 k_1}/n}{(n_{k_0}/n)(n'_{k_1}/n)}$$

Variation of information (VI) is defined as following:

$$VI(\mathcal{C}, \mathcal{C}') = \mathcal{H}(\mathcal{C}) + \mathcal{H}(\mathcal{C}') - \mathcal{I}(\mathcal{C}, \mathcal{C}') = \mathcal{H}(\mathcal{C}|\mathcal{C}') + \mathcal{H}(\mathcal{C}'|\mathcal{C})$$

- Generalized F-measure

The recall and precision for any soft class $C'_{k_1} \in C'$ and cluster $C_{k_0} \in C$ are defined as follows:

$$\text{Recall}(C_{k_0}, C'_{k_1}) = \frac{\sum_i \min(p_{i,c_{k_0}}, p_{i,c'_{k_1}})}{\sum_i p_{i,c_{k_0}}}$$

$$\text{Precision}(C_{k_0}, C'_{k_1}) = \frac{\sum_i \min(p_{i,c_{k_0}}, p_{i,c'_{k_1}})}{\sum_i p_{i,c'_{k_1}}}$$

Then the F measure is:

$$F(\mathcal{C}', \mathcal{C}) = \sum_{k_0} \frac{n_{k_0}}{n} \max_{k_1} F(C_{k_0}, C'_{k_1})$$

The generalized F measure is still in [0,1]. $F(\mathcal{C}', \mathcal{C}) = 1$ if and only if $\mathcal{C}$ and $\mathcal{C}'$ are the same.

A.3 Measures for Hierarchical Clustering

We surveyed two kinds of measures for hierarchical clustering. One way is to directly compare the whole clustering tree, while another is to compare level by level. The limitation of those measures is that they only apply to the dendrogram tree, i.e., a tree with $|V| - 1$ levels and number of cluster decreases by 1 between each neighboring level. More generally speaking, they only apply to the case that two hierarchical clustering have the same number of levels and the same numbers of clusters in each level.

Given data $\mathcal{D}$ with n data points, let H_1 and H_2 be two dendrograms, we now describe these two kinds of measures.

A.3.1 Whole Dendrogram Measure

We firstly need a mathematical way to represent the whole dendrograms. Dissimilarity matrix is introduced for this purpose. Dissimilarity matrix M is an $n \times n$ symmetric matrix, in which each index corresponds to a data point.

Two approaches have been employed to compute the elements of dissimilarity matrix corresponding to a given hierarchical clustering. [33] proposed to set $M_{ij} = M_{ji}$ equal to the

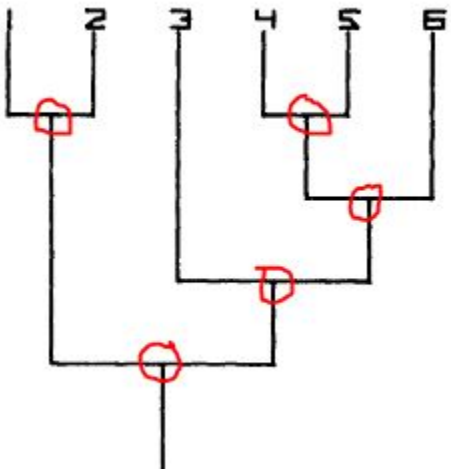

Figure A.3.1: Points between node 1 and 3 in dendrogram

level in which i and j are firstly merged to the same cluster. The diagonal of M is set as 0.

A second way ([27]) is to set $M_{ij} = M_{ji}$ as the number of nodes of the dendrogram between i and j. An example of M_{14}in a dendrogram is given in figure 5.1, in which the 5 circled nodes are between node 1 and 4. Therefore, $M_{14} = 5$ in this case.

One can easily see that the dendrogram can be uniquely recovered from any of these two kinds of dissimilarity matrix.

After we have the dissimilarity matrix, say $M_1$and M_2, for two dendrogram H_1 and H_2. The rest of the task is to find ways to compare these two matrices. Generally there are many ways and some are listed as follows:

$$\Delta = \sum |M_1 - M_2| \tag{A.1}$$

$$\Delta_\mu = (\sum |M_1 - M_2|^{1/\mu})^\mu / (\sum |M_1|^{1/\mu})^\mu \tag{A.2}$$

$$r = \sum (D_1 - \bar{D_1})(D_2 - \bar{D_2}) / [\sum (D_1 - \bar{D_1})^2 \sum (D_2 - \bar{D_2})^2]^{1/2} \tag{A.3}$$

For more matrix comparison measures, one may refer to the Table 5 in [29].

A.3.2 Level-wise Measure

Comparing hierarchical clustering by levels seems more straightforward. The basic idea in [10]is to plot the point $(k, \phi(k))$ for $k = 1, ..., n-1$, in which $\phi(k)$ is a measure describing the distance between clustering at level k.

Generally, $\phi(k)$ can be any measure described in subsection 5.2. [10] also proposed a new measure for $\phi(k)$:

$$\phi(k) = T_k / \sqrt{P_k Q_k}$$

where $T_k = \sum_{i=1}^{k} \sum_{j=1}^{k} n_{ij}^2 - n$, $P_k = \sum_{i=1}^{k} n_i^2 - n$ (for H_1) and $Q_k = \sum_{i=1}^{k} (n_i')^2 - n$ (for H_2).

Note that in the above formula, n_{ij} is the number of shared points between cluster i of H_1 and cluster j of H_2 at level k.

The $\phi(k)$ defined above has some good properties. $\phi(k) \in [0, 1]$. $\phi(k) = 1$ if and only if the clusters of $H_1$and H_2 at level k are identical. $\phi(k) = 0$ when each n_{ij} is 0 or 1, so that every pair of points that appear in the same cluster in $H_1$are assigned to different clusters in H_2. Furthermore, T_k has a good interpretation:

$$T_k = \sum_{i=1}^{k} \sum_{j=1}^{k} n_{ij}^2 - n = \sum_{i=1}^{k} \sum_{j=1}^{k} n_{ij}^2 - \sum_{i=1}^{k} \sum_{j=1}^{k} n_{ij} = 2 \sum_{i=1}^{k} \sum_{j=1}^{k} \binom{n_{ij}}{2}$$

which is the number of point pairs that are in the same cluster both in H_1 and H_2.

References

[1] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, May 2009.

[2] Amir Ben-dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. pages 49–57, 2002.

[3] A. Brandt, S. F. Mccormick, and J. W. Ruge. Sparsity and its Applications; Algebraic multigrid (AMG) for sparse matrix equations. Cambridge University Press, New York, 1984.

[4] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[5] Jean-Philippe P. Brunet, Pablo Tamayo, Todd R. Golub, and Jill P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences of the United States of America*, 101(12):4164–4169, March 2004.

[6] Y. Cheng and G. M. Church. Biclustering of expression data. *International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology*, 8:93–103, 2000.

[7] Leon Danon, Albert D. Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(9):P09008–09008, September 2005.

[8] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference*

on Knowledge discovery and data mining, KDD '01, pages 269–274, New York, NY, USA, 2001. ACM.

[9] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 89–98, New York, NY, USA, 2003. ACM.

[10] E. B. Fowlkes and C. L. Mallows. A Method for Comparing Two Hierarchical Clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[11] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[12] HOLLERITH HERMAN. Radix sort, 1887.

[13] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the Yeast transcriptional network. *Nature genetics*, 31(4):370–377, August 2002.

[14] Tiffany Inglis, Hans De Sterck, Geoffrey Sanders, Haig Djambazian, Robert Sladek, Saravanan Sundararajan, and Thomas J. Hudson. Multilevel space-time aggregation for bright field cell microscopy segmentation and tracking. *Journal of Biomedical Imaging*, 2010:8:1–8:21, January 2010.

[15] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral, 2000.

[16] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, June 2007.

[17] Jingu Kim and Haesun Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 353–362, Washington, DC, USA, 2008. IEEE Computer Society.

[18] Dan Kushnir, Meirav Galun, and Achi Brandt. Fast multiscale clustering and manifold identification. *Pattern Recogn.*, 39(10):1876–1891, October 2006.

[19] Dan Kushnir, Meirav Galun, and Achi Brandt. Fast multiscale clustering and manifold identification. *Pattern Recogn.*, 39(10):1876–1891, October 2006.

[20] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[21] Aaron F. McDaid, Derek Greene, and Neil Hurley. Normalized Mutual Information to evaluate overlapping community finding algorithms. 2011.

[22] Marina Meilă. Comparing clusterings—an information based distance. *J. Multivar. Anal.*, 98(5):873–895, May 2007.

[23] Marina Meilă and David Heckerman. An experimental comparison of model-based clustering methods. *Mach. Learn.*, 42(1-2):9–29, January 2001.

[24] T. M. Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pac. Symp. Biocomput*, pages 77–88, 2003.

[25] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

[26] Hossein Parsaei. Finding the number of clusters in a dataset: A review and simulation study. 2013.

[27] J. B. Phipps. Dendrogram Topology. *Systematic Zoology*, 20(3), 1971.

[28] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, May 2006.

[29] F. James Rohlf. Methods of comparing classifications. *Annual Review of Ecology and Systematics*, 5(1):101–113, 1974.

[30] Eitan Sharon, Achi Brandt, and Ronen Basri. Fast multiscale image segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 70–77, 1999.

[31] Eitan Sharon, Meirav Galun, Dahlia Sharon, Ronen Basri, and Achi Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, June 2006.

[32] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 731–, Washington, DC, USA, 1997. IEEE Computer Society.

[33] Robert R. Sokal and F. James Rohlf. The Comparison of Dendrograms by Objective Methods. *Taxon*, 11, 1962.

[34] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[35] Hans De Sterck. A self-learning algebraic multigrid method for extremal singular triplets and eigenpairs. *CoRR*, abs/1102.0919, 2011.

[36] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. In *In Proceedings of ISMB 2002*, pages 136–144, 2002.

[37] Stephen A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, page 2009, 2009.

[38] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 267–273, New York, NY, USA, 2003. ACM.

[39] Raphael Yuster and Uri Zwick. Fast sparse matrix multiplication. *ACM Trans. Algorithms*, 1(1):2–13, July 2005.